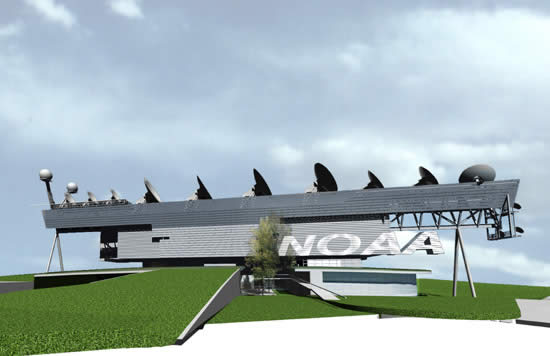
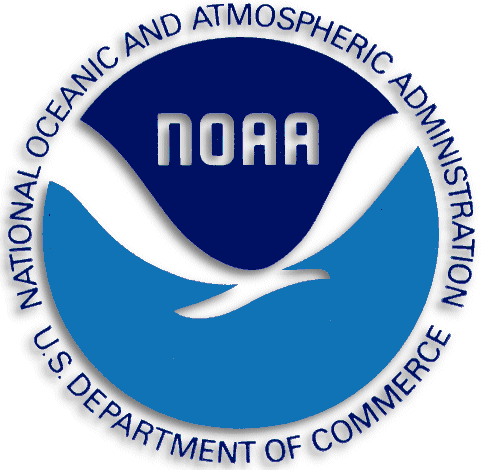
**Environmental Satellite Processing Center (ESPC)**

****

**NOAA Unique CrlS/ATMS Product System (NUCAPS)**

**System Maintenance Manual (SMM)**

**Version 4.1, June 8, 2015**

****

**Version 1.0 prepared by: Thomas S. King, Dell & Chen Zhang**

**Version 2.0 Modified by: Letitia Soulliard and Thomas King**

**Version 3.0 Updated by: Awdesh Sharma, NOAA/NESDIS, Oleg Roytburd, SGT**

**Version 4.0 Modified by: Letitia Soulliard and Thomas King**

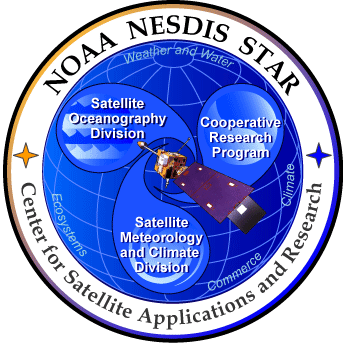
**U.S. Department of Commerce**

**National Oceanic and Atmospheric Administration (NOAA)**

**National Environmental Satellite, Data, and Information Service (NESDIS)**

**Office of Satellite Products and Operations (OSPO)**

**Environmental Satellite Processing Center (ESPC)**



NOAA NESDIS

CENTER for SATELLITE APPLICATIONS and RESEARCH

The NOAA Unique CrIS/ATMS Product System

System Maintenance Manual

Approval Page

|  |  |  |
| --- | --- | --- |
|  | |  |
| Environmental Satellite Processing Center (ESPC)  The NOAA Unique CrlS/ATMS Product System  System Maintenance Manual (SMM)  Version 4.1  WR000224 | | |
| **PROGRAM**: Atmosphere | | DOCUMENT RELEASE DATE: June 2015 |
| APPROVALS | | |
| GROUP: ESPC Date  eApproval  NAME: A.K. Sharma, Program Area Lead | GROUP: ESPC Date  eApproval  NAME: Oleg Roytburd, Programmer | |
| GROUP: ESPC Date  eApproval  NAME: Lindsley Bodden, Contract Manager | GROUP: ESPC Date | |

**Document History**

The Document History identifies the document origination and the series of revisions to the document that may occur at any point after the baseline release. This page will become a permanent part of this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version  Number | Date | Description of Change/Revision | Section/Pages  Affected | Changes Made  by Name/Title/ Organization |
| 1.0 | 03/14/2012 | Origination; Document was released for Phase 1 SDR products. | All | Chen Zhang (Dell)  Thomas S King (IMSG)  Letitia Soulliard (IMSG) |
| 2.0 | 9/27/2012 | The entire document was revised for Phase 2 EDR products | All | Letitia Soulliard and Thomas King |
| 3.0 | 05/29/2014 | Modified by A.K. Sharma | All |  |
| 3.0 | 06/12/2014 | Technical Edit | All |  |
| 3.0 | 6/12/2014 | Quality Assessment by Marilyn Gross | All | 3.0 |
| 3.0 | 6/17/2014 | Minor Edits | 1.1.3 | 3.0 |
| 3.0 | 07/01/2014 | Quality Assessment by Janice Olfus-Carter | All | 3.0 |
| 4.0 | 6/8/2015 | Updated for Phase 3 | All | Letitia Soulliard and Thomas King |
| 4.1 | 9/10/2015 | Updated for Phase 3 | All | Thomas King |

TABLE OF CONTENTS

[TABLE OF CONTENTS vii](#_Toc430079320)

[List of Tables xi](#_Toc430079321)

[List of Figures xii](#_Toc430079322)

[LIST OF ACRONYMS xiii](#_Toc430079323)

[EXECUTIVE SUMMARY 1](#_Toc430079324)

[1.0 INTRODUCTION 9](#_Toc430079325)

[1.1. Product Overview 9](#_Toc430079326)

[1.2. Algorithm Overview 10](#_Toc430079327)

[1.3. Interfaces Overview 11](#_Toc430079328)

[2. HARDWARE 11](#_Toc430079329)

[2.1. Hardware Description 11](#_Toc430079330)

[2.2. Operating System 13](#_Toc430079331)

[2.3. System Requirements 14](#_Toc430079332)

[2.3.1. Storage Requirements 14](#_Toc430079333)

[2.3.2. Computer Resource Requirements 15](#_Toc430079334)

[2.3.3. Communication Needs 17](#_Toc430079335)

[3. SOFTWARE 21](#_Toc430079336)

[3.1. Software Description 21](#_Toc430079337)

[3.2. Directory Description 23](#_Toc430079338)

[3.3. Source Code Description 28](#_Toc430079339)

[4. NORMAL OPERATIONS 35](#_Toc430079340)

[4.1. System Control 35](#_Toc430079341)

[4.1.1. System Control Files 35](#_Toc430079342)

[4.1.2. Processing Controls 36](#_Toc430079343)

[4.2. Installation 59](#_Toc430079344)

[4.2.1. Setup Requirements 59](#_Toc430079345)

[4.2.2. Installation Procedures 61](#_Toc430079346)

[4.3. Build 65](#_Toc430079347)

[4.3.1. Compilation Procedures 65](#_Toc430079348)

[4.3.2. Build Procedure 66](#_Toc430079349)

[4.4. Operations Procedures 66](#_Toc430079350)

[4.4.1. Normal Operations 66](#_Toc430079351)

[4.4.2. Data Preparation 75](#_Toc430079352)

[4.5. Distribution 76](#_Toc430079353)

[4.5.1. Data Transfer / Communications 76](#_Toc430079354)

[4.5.2. Distribution Restrictions 77](#_Toc430079355)

[4.5.3. Product Retention Requirements 77](#_Toc430079356)

[4.5.4. External Product Tools 77](#_Toc430079357)

[5. MONITORING AND MAINTENANCE 79](#_Toc430079358)

[5.1. Job Monitoring 79](#_Toc430079359)

[5.2. NUCAPS Quality Assurance 79](#_Toc430079360)

[5.3. Product Monitoring 80](#_Toc430079361)

[5.3.1. Unit Test Plans 80](#_Toc430079362)

[5.3.2. Internal Product Tools 80](#_Toc430079363)

[5.3.3. Performance Statistics 81](#_Toc430079364)

[5.3.4. Product Monitoring 83](#_Toc430079365)

[5.3.5. Product Criticality 84](#_Toc430079366)

[5.4. Maintenance 84](#_Toc430079367)

[5.4.1. Monitoring 84](#_Toc430079368)

[5.4.2. Science Maintenance 96](#_Toc430079369)

[5.4.3. Library Maintenance 106](#_Toc430079370)

[5.4.4. Special Procedures 106](#_Toc430079371)

[5.4.5. Maintenance Utilities 106](#_Toc430079372)

[5.5. Program Backup 107](#_Toc430079373)

[6. TROUBLESHOOTING 109](#_Toc430079374)

[6.1. Problem Diagnosis and Recovery 109](#_Toc430079375)

[6.1.1. Quality Control Output 109](#_Toc430079376)

[6.1.2. Error Correction 110](#_Toc430079377)

[6.1.3. Problem Diagnosis and Recovery Procedures 126](#_Toc430079378)

[6.1.4. Data Recovery Procedures 135](#_Toc430079379)

[6.1.5. Program Recovery Procedures 135](#_Toc430079380)

[6.2. Application Shutdown and Restart 136](#_Toc430079381)

[6.2.1. Application Shutdown Procedures 136](#_Toc430079382)

[6.2.2. Application Restart Procedures 137](#_Toc430079383)

[6.3. System Shutdown and Restart 139](#_Toc430079384)

[6.3.1. System Shutdown Procedures 139](#_Toc430079385)

[6.3.2. System Restart Procedures 139](#_Toc430079386)

[6.3.3. System Reboot Procedures 139](#_Toc430079387)

[7. APPENDIX 141](#_Toc430079388)

[7.1. Data Flow 141](#_Toc430079389)

[7.1.1. External Interfaces 141](#_Toc430079390)

[7.1.2. System Level Design and Data Flow 146](#_Toc430079391)

[7.1.3. Unit Level Design and Data Flow 185](#_Toc430079392)

[7.2. Input Data Files 198](#_Toc430079393)

[7.3. Ancillary Data Files 201](#_Toc430079394)

[7.3.1. Digital Elevation Model 201](#_Toc430079395)

[7.3.2. Retrieval System Files 201](#_Toc430079396)

[7.3.3. GFS Forecast Files 203](#_Toc430079397)

[7.3.4. Eigenvector Files 205](#_Toc430079398)

[7.3.5. OLR Boxcar files 206](#_Toc430079399)

[7.3.6. VIIRS collocation LUT Files 207](#_Toc430079400)

[7.3.7. Template Files 207](#_Toc430079401)

[7.4. Reference Data Files 210](#_Toc430079402)

[7.5. Intermediate Data Files 210](#_Toc430079403)

[7.5.1. NetCDF Files 210](#_Toc430079413)

[7.5.2. Retrieval Input/Output Files 214](#_Toc430079414)

[7.6. Output Data Files 216](#_Toc430079415)

[7.6.1. Global Grids 216](#_Toc430079416)

[7.6.2. Global Binaries 219](#_Toc430079417)

[7.6.3. Global Matchups 220](#_Toc430079418)

[7.6.4. EDR Files 221](#_Toc430079419)

[7.6.5. Quality Monitoring Files 222](#_Toc430079420)

[7.7. Archive Data Files 222](#_Toc430079421)

[8. List of References 225](#_Toc430079422)

List of Tables

Page

[Table 0‑1 NUCAPS Product Team 1](#_Toc430079423)

[Table 2‑1 NDE HW Layout & Flow 11](#_Toc430079424)

[Table 2‑2 NDE Storage Requirements 14](#_Toc430079425)

[Table 2‑3 NUCAPS System Processing Requirements 15](#_Toc430079426)

[Table 2‑4 NUCAPS Input Data Sizing Per Provider 18](#_Toc430079427)

[Table 2‑5 NUCAPS Output Products Sizing Per User Per Day 18](#_Toc430079428)

[Table 5‑1 Production Job Spec States 87](#_Toc430079429)

[Table 5‑2 Production Job States 88](#_Toc430079430)

[Table 5‑3 Distribution Prepare Request States 93](#_Toc430079431)

[Table 5‑4 Pull Distribution Job States 94](#_Toc430079432)

[Table 5‑5 Push Distribution Job States 96](#_Toc430079433)

[Table 5‑6 Statistics Generation 97](#_Toc430079434)

[Table 6‑1 DHS Error Messages 111](#_Toc430079435)

[Table 6‑2 NUCAPS Error Messages (High-Level) 128](#_Toc430079436)

[Table 6‑3 NUCAPS Error Messages (Low-Level) 131](#_Toc430079437)

[Table 6‑4 Retrieval Error Codes 134](#_Toc430079438)

[Table 7‑1 NUCAPS External Files 145](#_Toc430079439)

[Table 7‑2 Preprocessor Unit Data Flow 149](#_Toc430079440)

[Table 7‑3 Subsetter Unit Data Flow 163](#_Toc430079441)

[Table 7‑4 Retrieval Unit Data Flows 169](#_Toc430079442)

[Table 7‑5 Validation Unit Data Flow 176](#_Toc430079443)

[Table 7‑6 Retrieval System Files 201](#_Toc430079444)

[Table 7‑7 NUCAPS CDL Files 208](#_Toc430079445)

List of Figures

Page

[Figure 0‑1 NDE Hardware Layout and Flow 5](#_Toc430079446)

[Figure 0‑2 NDE System Diagram 6](#_Toc430079447)

[Figure 0‑3 NUCAPS External Interfaces 8](#_Toc430079448)

[Figure 0‑4 NDE Data Access Process 9](#_Toc430079449)

[Figure 5‑1-1 NDE DHS Dashboard GUI Page 85](#_Toc430079450)

[Figure 5‑2 NDE DHS Backlog and Throughput GUI Page 86](#_Toc430079451)

[Figure 5‑3 NDE DHS PGS Status Summary GUI Page 91](#_Toc430079452)

[Figure 5‑4 NDE DHS PGS Job Spec & Job Status GUI Page 92](#_Toc430079453)

[Figure 5‑5 NDE DHS PGS Job Spec & Job Status Details GUI Page 93](#_Toc430079454)

[Figure 7‑1 NUCAPS External Interfaces 142](#_Toc430079455)

[Figure 7‑2 NUCAPS Context Level Data Flow 144](#_Toc430079456)

[Figure 7‑3 NUCAPS System Level Data Flow 148](#_Toc430079457)

[Figure 7‑4 Preprocessor Unit Data Flow 187](#_Toc430079458)

[Figure 7‑5 Collocator Sub-Unit Data Flow 188](#_Toc430079459)

[Figure 7‑6 Subsetter Unit Data Flow 190](#_Toc430079460)

[Figure 7‑7 Retrieval Unit Data Flow 192](#_Toc430079461)

[Figure 7‑8 Validation Unit Data Flow - Part 1 196](#_Toc430079462)

[Figure 7‑9 Validation Unit Data Flow - Part 2 197](#_Toc430079463)

[Figure 7‑10 Validation Unit Data Flow - Part 3 198](#_Toc430079464)

LIST OF ACRONYMS

|  |  |
| --- | --- |
| AFWA | Air Force Weather Agency |
| AMSU-A | Advanced Microwave Sounder Unit - A |
| ASCII | American Standard Code for Information Interchange |
| ATBD | Algorithm Theoretical Basis Document |
| ATMS | Advanced Technology Microwave Sounder |
| BUFR | Binary Universal Form for the Representation of meteorological data |
| CCR | Cloud-Cleared Radiances |
| CDL | Common Data Language |
| CDR | Critical Design Review |
| CLASS | Comprehensive Large Array-data Stewardship System |
| CPU | Central Processing Unit |
| CrIS | Cross-track Infrared Sounder |
| DAP | Delivered Algorithm Package |
| DEM | Digital Elevation Model |
| DDS | Data Distribution Server |
| DHS | Data Handling System |
| DOD | Department of Defense |
| EDR | Environmental Data Record |
| EPL | Enterprise Product Lifecycle |
| ESPC | Environmental Satellite Processing Center |
| EUMETSAT | European Organization for the Exploitation Meteorological Satellites |
| FOR | Field Of Regard |
| FOV | Field of View |
| GB | Gigabyte |
| GFS | Global Forecast System |
| GMAO | Global Modeling and Assimilation Office |
| GMT | Greenwich Mean Time |
| GRIB | Gridded Binary format |
| IASI | Infrared Atmospheric Sounding Interferometer |
| ICD | Interface Control Document |
| IDPS | Interface Data Processing Segment |
| IP | Intermediate Product |
| IPD | NOAA’s Internal Processing Division |
| IPT | Integrated Product Team |
| NDE | NPOESS Data Exploitation |
| NGDC | National Geophysical Data Center |
| NCDC | National Climate Data Center |
| NCEP | National Center for Environmental Prediction |
| NESDIS | National Environmental Satellite, Data, and Information Service |
| netCDF4 | network Common Data Format version 4 |
| NOAA | National Oceanic and Atmospheric Administration |
| NPOESS | National Polar-orbiting Operational Environmental Satellite System |
| NRL | Naval Research Lab |
| NSOF | NOAA Satellite Operations Facility |
| NUCAPS | NOAA Unique CrIS/ATMS Product System |
| NWP | Numerical Weather Prediction |
| OLR | Outgoing Longwave Radiances |
| OSPO | Office of Satellite & Product Operations |
| PBR | Project Baseline Report |
| PCF | Process Control File |
| PCS | Principal Components |
| PGAI | Product Generation Application Interface |
| PGM | Product Generation Manager |
| PSF | Process Status File |
| RAD | Requirements Allocation Document |
| RR | Reconstructed Radiances |
| RSE | Remote Sensing Extension |
| SADIE | Science Algorithm Development and Integration Environment |
| SAN | Storage Area Network |
| SDR | Sensor Data Record |
| SMCD | Satellite Meteorology and Climate Division |
| SPSRB | Satellite Products and Services Review Board |
| STAR | Center for Satellite Applications and Research |
| SWA | Software Architecture Document |
| VIIRS | Visible Infrared Imager Radiometer Suite |
| VVP | Verification and Validation Plan |
| WMO | World Meteorological Organization |
| XML | eXtensible Markup Language |

EXECUTIVE SUMMARY

The NOAA Unique CrIS/ATMS Processing System (NUCAPS) generates (1) spectrally and spatially thinned radiances, (2) retrieved products such as profiles of temperature, moisture, OLR, and trace gases and cloud-cleared radiances, and (3) global validation products such as matchups and gridded radiances and profiles. The system was designed to run within the NPOESS Data Exploitation (NDE) production environment. The NDE production environment is, in turn, delivered to the Office of Satellite and Product Operations (OSPO) where it is run operationally. NUCAPS is identified as a mission critical and therefore a 24 X 7 service maintenance level is required. Return to service within 2 hours is required.

Requirements for the NUCAPS products are described in the NUCAPS Requirements Allocation Document (RAD) (NESDIS/STAR, 2011) which is available in the NUCAPS project artifact repository.

The NUCAPS product team consists of the following individuals:

Table 0‑1 NUCAPS Product Team

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Organization | Role | Contact Information |
| Walter Wolf | STAR | STAR Project Lead | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-1314  Email: [Walter.Wolf@noaa.gov](mailto:Walter.Wolf@noaa.gov) |
| Quanhua Liu | STAR | STAR Algorithm Lead | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3661  Email: Quanhua.Liu@noaa.gov |
| Changyi Tan | STAR | Retrieval Scientist | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3541  Email: [Changyi.Tan@noaa.gov](mailto:Changyi.Tan@noaa.gov) |
| Awdhesh Sharma | OSPO | PAL | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3229  Email: [Awdhesh.Sharma@noaa.gov](mailto:Awdhesh.Sharma@noaa.gov) |
| Thomas King | STAR | Development Lead | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3547  Email: [Thomas.S.King@noaa.gov](mailto:Thomas.S.King@noaa.gov) |
| Peter Keehn | STAR | Developer | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3548  Email: [Peter.Keehn@noaa.gov](mailto:Peter.Keehn@noaa.gov) |
| Letitia Soulliard | STAR | Developer | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3538  Email: Letitia.Soulliard@noaa.gov |
| Chris Barnet | STC | Retrieval Scientist | 10015 Old Columbia Road E-250  Columbia, MD 21046  Phone: 410-309-0818  Email: Barnet@stcnet.com |
| Antonia Gambacorta | STC | Retrieval Scientist | 10015 Old Columbia Road E-250  Columbia, MD 21046  Phone: 410-309-0818  Email: Antonia.Gambacorta@stcnet.com |
| Kexin Zhang | STAR | Quality Assurance Lead | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3544  Email: [Kexin.Zhang@noaa.gov](mailto:Kexin.Zhang@noaa.gov) |
| Larisa Koval | STAR | Requirements Developer | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3542  Email: [Larisa.Koval@noaa.gov](mailto:Larisa.Koval@noaa.gov) |
| Yunhui Zhao | STAR | Configuration Manager | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3543  Email: [Yunhui.Zhao@noaa.gov](mailto:Yunhui.Zhao@noaa.gov) |
| Oleg Roytburd | OSPO | Primary Maintenance Programmer | 5830 University Research Court  College Park, MD. 20740  Phone: 301-683-3283  Email: oleg.roytburd@noaa.gov |
| Robert Lindsay | OSPO | Secondary Maintenance Programmer | 5830 University Research Court.  College Park, Maryland 20740  Phone: 301-683-3281  Email: robert.lindsay@noaa.gov |
| William P. O’Connor | OSPO | Secondary Maintenance Programmer | 5830 University Research Court.  College Park, Maryland 20740 PHONE:  301-683-3280  FAX:    301-683-3297  E-MAIL: [william.oconnor@noaa.gov](mailto:william.oconnor@noaa.gov) |
| Tony Reale | STAR | STAR Scientist | NSOF  4231 Suitland Rd  Suitland MD 20746  Email: tony.reale@noaa.gov |
| Donna McNamara | OSPO | Data Distribution | NSOF 1372  4231 Suitland Rd  Suitland MD 20746  Phone: (301)817-3803  Email; donna.mcnamara@noaa.gov |

The NDE Data Handling System (DHS) consists of Ingest (INS), Product Generation (PGS), Product Distribution (PDS), and Monitoring subsystems; and is physically spread across numerous servers, both IBM and Linux based, and a Storage Area Network (SAN). The IBM Power 7 Series blade servers run the Advanced Interactive eXecutive (AIX) operating system while the Dell blade servers run Redhat Enterprise Linux (RHEL). All servers, i.e. nodes, are 64-bit. The Quantum StorNext SAN file system has a capacity of 60 TB and is used to manage all data storage. An Ethernet network is used as the front-end for all clients to access files on the SAN.

There are six AIX nodes dedicated to product generation, each with 8 quad core CPUs resulting in 32 effective CPUs and 63 GB of memory for each node. The Power 7 series CPUs have a clock frequency of 3 GHz. The Linux nodes have 24 Intel Xeon (x86\_64 architecture) CPUs running at 2.8 GHz clock frequency with 37 GB of memory each. The Linux nodes are spread across ingest (2), product generation (2 for the factories and 2 for processing), and product distribution (4 for pull and 4 for push) for a total of 14 nodes. Each processing node (both AIX and Linux) has a RAID subsystem providing local storage of 1.5 TB.

All NOAA Unique Product (NUP) algorithms currently run on the AIX processing nodes. There are plans to transition all NUP algorithms to Linux processing nodes and add more Linux processing capability in the future.

The following diagram shows the data flow through the NDE DHS and the physical layout of the hardware.



Figure ‑ NDE Hardware Layout and Flow

The following diagram shows the network layout for the NDE DHS Production Environment-1 (PE-1).



Figure ‑ NDE System Diagram

All low-level processing code in the NUCAPS is written in compiled languages.

These are Fortran 90, Fortran 77, C++, and C. Low-level code performs all data processing, scientific computation, read/writing, reformatting, and opening/closing of files. All high-level code is written in Perl. High-level tasks include file management, system management, making system calls, and error trapping from low-level processing. The Perl scripts act as driver code for the lower-level processing. The NUCAPS software is organized in to four processing units: (1) the *Preproc Unit* handles all front end processing, (2) the *Subsetter Unit* performs the radiance thinning, (3) the *Retrieval Unit* runs the retrieval and retrieval reformatting code, and (4) the *Validation Unit* generates the daily global datasets used for validation and quality monitoring. Each software unit is run by its own Perl driver script.

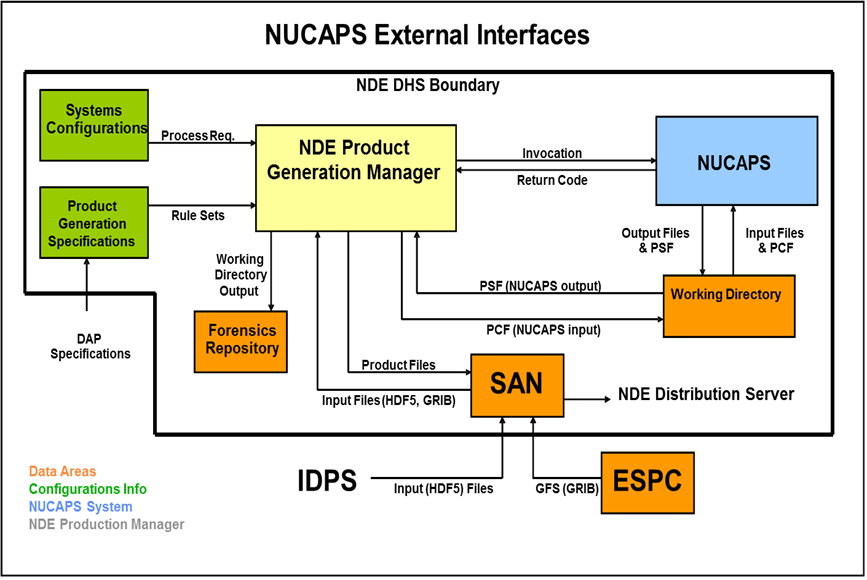


Figure ‑ NUCAPS External Interfaces

The NUCAPS profile products and cloud-cleared radiances are generated using a retrieval algorithm whereas the thinned radiances and global products do not require a science algorithm and can be conceived of as a reorganization of the data. The retrieval algorithm runs inside a system of supporting software. This system was developed during the Aqua mission to use data from the AIRS/AMSU/MODIS instruments, but was designed to be flexible to use IASI/AMSU-A/MHS/AVHRR and CrIS/ATMS/VIIRS. Therefore, even though it is referred to in this document as the NUCAPS algorithm, it is in fact identical to that used for AIRS and IASI. The NUCAPS retrieval algorithm has a flexible modular design that allows the types of instruments, the amount of diagnostics, and the activation of various retrieval process steps to be turned on or off via a set of input name-lists. This flexibility allows the system to be used for research or in a faster and more efficient operational manner. For information about the NUCAPS algorithm, see the NUCAP Algorithm Theoretical Basis Document (ATBD).

The NESDIS' Policy on Access and Distribution of Environmental Data and Products is provided at: <http://www.ospo.noaa.gov/Organization/About/access.html>.

Users need to fill out the Data Access Request Form located on this site and submit to the PAL with a copy to [nesdis.data.access@noaa.gov](mailto:nesdis.data.access@noaa.gov). This address provides the OSPO Data Access Team a copy of the correspondence. The process is defined in the following diagram. Once the request is approved by the OSPO management the data will be delivered by the Data Distribution System (DDSProd) currently distributing the ESPC data products and later by the Product Distribution and Access (PDA) system. The ESPC Data Distribution Manager, Donna McNamara ([donna.mcnamara@noaa.gov](mailto:donna.mcnamara@noaa.gov)) should be contacted for any data accessibility and data distribution problems.

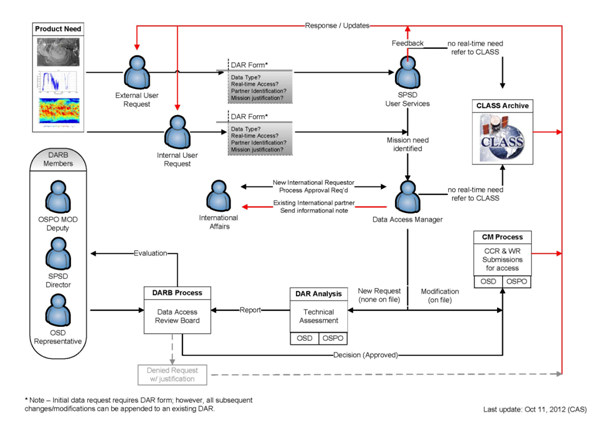


Figure ‑ NDE Data Access Process

# 1.0 INTRODUCTION

## Product Overview

The NOAA Unique CrIS/ATMS Processing System (NUCAPS) was developed to generate (1) spectrally and spatially thinned radiances, (2) retrieved products such as profiles of temperature, moisture, and trace gases and cloud-cleared radiances, and (3) global validation products such as matchups and gridded radiances and profiles. The system was designed to run within the NPOESS Data Exploitation (NDE) system delivered to the Office of Satellite and Product Operations (OSPO). The output products are intended for operational and scientific users.

## Algorithm Overview

The NUCAPS profile products and cloud-cleared radiances are generated using a retrieval algorithm whereas the thinned radiances and global products do not require a science algorithm and can be conceived of as a reorganization of the data. The retrieval algorithm runs inside a system of supporting software. This system was developed during the Aqua mission to use data from the AIRS/AMSU/MODIS instruments, but was designed to be flexible to use IASI/AMSU-A/MHS/AVHRR and CrIS/ATMS/VIIRS. Therefore, even though it is referred to in this document as the NUCAPS algorithm, it is in fact identical to that used for AIRS and IASI. The NUCAPS retrieval algorithm has a flexible modular design that allows the types of instruments, the amount of diagnostics, and the activation of various retrieval process steps to be turned on or off via a set of input name-lists. This flexibility allows the system to be used for research or in a faster and more efficient operational manner. For information about the NUCAPS algorithm, see the NUCAP Algorithm Theoretical Basis Document (NESDIS/STAR, 2009).

## Interfaces Overview

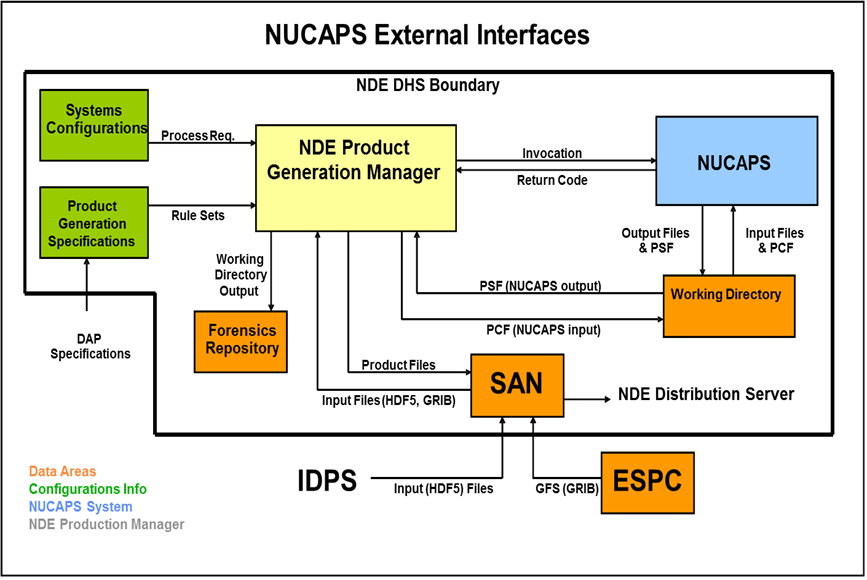


Figure 1.1 NUCAPS External Interfaces

# HARDWARE

## Hardware Description

The NDE Data Handling System (DHS) consists of Ingest (INS), Product Generation (PGS), Product Distribution (PDS), and Monitoring subsystems; and is physically spread across numerous servers, both IBM and Linux based, and a Storage Area Network (SAN). All servers, i.e. nodes, are 64-bit. An Ethernet network is used as the front-end for all nodes to access files on the SAN. The following table lists all pertinent hardware items and their NDE function.

Table 2‑1 NDE HW Layout & Flow

|  |  |
| --- | --- |
| Hardware Item | NDE Function |
| IBM BladeCenter H Chassis |  |
| (7) IBM PS701 Blade Servers | Product Generation, Science & Tailoring Alg |
| (2) Cisco 4GB Fibre Channel Switches | SAN/DAS connectivity for PS701 servers |
| (2) Cisco Ethernet Switches | Network connectivity for PS701 servers |
| IBM DS3512 Storage Array | Local storage for PS701 servers |
| IBM EXP3512 Expansion Array | Expansion chassis for IBM DS3512 Storage |
| (2) Dell PowerEdge R610 Servers | Oracle Database Servers |
| EMC AX4-5F Storage Array | Local Storage for R610 servers |
| Dell M1000E BladeCenter Chassis |  |
| (8) Dell M610 Blade Servers | Ingest, Tailoring, Distribution |
| (2) Dell Fibre Channel Modules | SAN/DAS connectivity for M610 servers |
| (2) Cisco 3130G Ethernet Switches | Network connectivity for M610 servers |
| EMC AX4-5F Storage Array | Local storage for M610 servers |
| (2) CAI Networks 690PG WebMuxes | FTP server load balancing |
| EMC Storage Array [ESPC controlled] | Shared storage |
| (2) Cisco MDS 9506 Fibre Channel Switches | SAN infrastructure component |
| (2) Cisco MDS 9222 Fibre Channel Switches | SAN infrastructure component |
| (2) Dell PowerEdge 2950 Servers | StorNext MDC servers [shared filesystem controller] |
| LSI Storage Array | Local storage for 2950 servers |
| (2) Cisco ASA 5500 Series Firewalls | Network boundary control [IDPS StorNext interface] |
| Cisco Catalyst 2960 Ethernet switch | StorNext network switch [shared network switch] |

## Operating System

The NDE Data Handling System (DHS) is implemented across a host of servers running both Advanced Interactive eXecutive (AIX) on Power Series 7 machines and Redhat Enterprise Linux on Dell Blade servers, both operating systems are based on 64 bit architectures. NOAA Unique Product (NUP) algorithms are compiled for the Linux operating system using the gnu compilers. DHS uses the open source JBoss Service-Oriented Architecture (SOA) platform to define and maintain all necessary services. An Oracle Database Management System provides the back-end for all data management. Each processing node (i.e. server) have science libraries installed locally. NDE maintains a set of approved libraries that algorithms can utilize: Hierarchical Data Format 5 (HDF5) and associated libraries (SZIP, ZLIB), Network Common Data Format (NetCDF-4), python, Perl, and Grib/Grib2.

## System Requirements

### Storage Requirements

Table 2-2 contains a list of storage requirements for all NUCAPS primary and ancillary input data, intermediate data, and output data. These requirements reflect the current build of NUCAPS.

Table 2‑2 NDE Storage Requirements

|  |  |
| --- | --- |
| Storage Item | Size (5 day data storage) |
| Data Products Files | 498 GB |
| Intermediate Files | 702 GB |
| Incoming NUCAPS Data | 368 GB |
| GFS Forecasts | 1.2 GB |
| System Code (executable) | 160 MB |
| System scripts (interpreted) | 1 MB |
| System Resource Files | 3.1 GB |
| Tools (library/utils) | 92 MB |
| Total | 1.57 TB |

Table 2‑2 also contains the storage requirements for all NUCAPS resource files, tools and system code. Tools such as IBM compilers, Perl, or the Gnu utilities are installed and managed by IT system administrators, and therefore, are not included in the table. Tools, such as NetCDF4, HDF5, and szlib, are required by NUCAPS, but are not considered to be part of the delivery since they are an NDE managed and shared resource. However, the storage for those libraries is included in the table. Note that almost all of the required storage consists of data volumes. The table assumes that the most recent five days of data are to be retained within the system.

### Computer Resource Requirements

This section describes the computer hardware and software resources needed for the operations environment. This includes storage capacity and timeliness requirements.

#### Processing Requirements

The processing requirement for all NUCAPS products is for them to be made available to users (e.g. on the distribution server) within three hours of observation. This requirement is identified in the NUCAPS RAD. The NUCAPS software was developed and tested to run on the RedHat Enterprise Linux on Dell Blade servers. Observation of run times shows that for nominal operational processing, the system requires an average of 4 CPUs at any given time to maintain latency. More CPUs will be required to keep up with the processing if the system has been shut down and is required to catch up.

Table 2‑3 lists the maximum memory and time requirements for each processing unit. These specifications on these processes assume nominal data flows. For product generation, the processing is assumed to be occurring on a single NUCAPS granule. The maximum memory is defined as the largest amount of memory that a given process uses at any time during its execution. The time shown is the amount of time required for the entire process to run, not the amount of time of peak memory consumption. The memory values are approximate since most data are dynamically allocated. Times are also approximate since they may vary due to variations in input/output load on the SAN and due to fluctuations in bandwidth availability to external servers.

NUCAPS granules arrive in the system at a rate of approximately 2 granules every minute and it takes about 1.5 minutes to process a granule from ingest through level 2 EDR distribution. At any given time there are typically between 3 and 16 granules being processed.

Table 2‑3 NUCAPS System Processing Requirements

|  |  |  |
| --- | --- | --- |
| Process | Max Memory | Time |
| NUCAPS\_Preproc.pl | ~18 MB | ~58 seconds |
| NUCAPS\_Subset.pl | ~26 MB | ~22 seconds |
| NUCAPS\_Retrieval.pl | ~360 MB | ~36 seconds |
| NUCAPS\_Validation.pl | ~120 MB | ~2 minutes |

#### Libraries and Utilities

NUCAPS requires the following libraries and utilities:

netCDF4 version 4.1.3

HDF5 version 1.8.7

szlib 2.1

wgrib2 v0.1.9.2a

uuid\_gen

These will be maintained by NDE or OSPO as they are a common resource shared by all the NDE product teams. NUCAPS anticipates regularly updating to the newest working and stable versions of these tools throughout the project lifecycle.

#### GNU Compilers

Only the test machine requires the presence of compilers. The production machine doesn’t need a compiler because only the tested compiled code is necessary on the production machine. After the system has been successfully tested on the test machine, it is moved (copied) over to the production machine. However, only the compiled code is moved over. This also ensures that nothing is changed after the test and that the identical code is used on the test machine in the event of a failover from production.

To compile on the test machine the GNU version 4.1.2 or higher C/C++ and Fortran 77/90 compiler must be present.

#### Perl

The system was developed using Perl version 5.8.8, but should function with any newer version. The following Perl modules are required by the system:

FileHandle

Fcntl

POSIX

Sys

File

Time

Net

Mail

These are all standard Perl library modules and shouldn’t vary significantly among versions of Perl. Perl is assumed to be at:

/bin/perl

### Communication Needs

Table 2‑4 shows the data volumes to be transferred to NDE. Table 2‑5 shows the data volumes to be transferred out of NDE’s data distribution server to the outside world on a user basis. These tables should be used as a guide for determining the disk space and bandwidth for the DHS.

Table 2‑4 NUCAPS Input Data Sizing Per Provider

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Number of Files/Day | Size/Day | Provider |
| CrIS GEO HDF5 | 2700 | 290 MB | IDPS |
| CrIS SDR HDF5 | 2700 | 47 GB | IDPS |
| ATMS GEO HDF5 | 2700 | 421 MB | IDPS |
| ATMS TDR HDF5 | 2700 | 420 MB | IDPS |
| VIIRS CH GEO HDF5 | 1012 | 566 MB | IDPS |
| VIIRS CH SDR HDF5 | 1012 | 379 MB | IDPS |
| VIIRS CF GEO HDF5 | 1012 | 23.4 GB | IDPS |
| VIIRS CF SDR HDF5 | 1012 | 1.38 GB | IDPS |
| GFS GRIB2 | 12 | 244 MB | NCEP |
| Total | 10820 | 74.1 GB | IDPS and NCEP |

Table 2‑5 NUCAPS Output Products Sizing Per User Per Day

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Number of Files/Day | Size/Day | User |
| NUCAPS ALL FOVs | 2700 | 16.1 GB | BUFR toolkit, OSPO |
| NUCAPS 399 ALL FOVs | 2700 | 5.3 GB | BUFR toolkit |
| NUCAPS PCS Monitoring | 2700 | 4 MB | OSPO |
| NUCAPS Retrieval Monitoring | 2700 | 245 MB | OSPO |
| EDR NetCDF | 2700 | 8.3 GB | CLASS and OSPO |
| CCR NetCDF | 2700 | 2.2 GB | CLASS |
| CCR Archive NetCDF | 2700 | 1.8 GB | CLASS |
| 0.5X2 NUCAPS EDR global grids | 2 | 1,450 MB | STAR |
| 0.5X2 OLR global grids | 2 | 5.8 MB | STAR |
| Total | 32419 | 35.4 GB |  |

# SOFTWARE

This section describes the system-level software elements of the system that are invoked by the NDE production system. For more detail about the entire NUCAPS design down to the unit-level and data flows at each level, see the Appendix. Next, this section describes the source code and system files delivered to NDE. These files are organized into subdirectories. The contents of each subdirectory are identified and their purpose is explained. Finally, the source code for each main program (Fortran 90) is identified and its function is explained.

## Software Description

There are four processing units in NUCAPS: (1) the Preprocessor unit, (2) the Subsetter unit, (3) the Retrieval unit, and (4) the Validation unit. Each unit is operated by its own Perl driver script.

The Preprocessor unit performs several important functions.

* It collocates a given set of CrIS, ATMS, and VIIRS granules and packages them into a single netCDF4 granule file. This will be done such that each set of array elements containing data from a CrIS FOR will be ordered to match (in space and time) to the associated arrays of ATMS and VIIRS data. This will allow data from the three instruments to be used easily together.
* It adds the topography (surface elevation) and land-sea-coast flag to the CrIS FOVs. This is derived from a Digital Elevation Model (DEM).
* It applies a Hamming apodization to the three CrIS radiance bands.
* It runs the OLR code to create the OLR netCDF4 output files.

The Subsetter unit resides downstream of the Preprocessor unit. It reads a given full resolution CrIS/ATMS/VIIRS netCDF4 file and thins its contents to produce a single set of netCDF4 output product files. Some of these files are used downstream by the Validation unit. Other files are directed by the DHS for tailoring into BUFR product files by the NDE tailoring tool suite.

The Retrieval unit uses the CrIS and ATMS data to produce trace gas products, surface and cloud properties, and cloud-cleared CrIS radiances. The Retrieval unit Perl script is triggered by the PGM when a CrIS/ATMS/VIIRS full resolution file is available to the DHS by the Preprocessor unit. Also, a GFS six hour forecast standard product file must also be available for the run period covered by the CrIS/ATMS/VIIRS file. The script processes only one granule. Multiple instances of the script are required to process multiple granules. As with the other units, the Perl script obtains its PCF from the working directory.

The Validation unit data flow is shown in Figure 7‑8, Figure 7‑9, and Figure 7‑10. The entire unit is run by a single Perl script indicated by the surrounding black box. However, due to the amount of content in the flow diagram, this box has been spread over the three figures. All the blue boxes are single Fortran main programs designed to produce a single type of product. Within the unit these Fortran programs run in series and not simultaneously even though most of them are not dependent on one another. These programs are consumptive of memory and CPU cycles and it is therefore better to run them in series. Furthermore, these are not products for near real-time use so they do not have the same timeliness constraints.

There are three basic types of validation products: grids, binaries, and matchups. These are daily products that are global in scope and coverage. The grids and matchups are products in which satellite observations are collocated to a set of pre-defined grid points (in the case of the gridded products), or to in-situ measurement locations and times (in the case of the matchup products). The binaries are just daily global compilations of the satellite data extracted from the granules. These can be thinned or full spatial resolution. The binary products are generated for intermediate processing or off-line use. For each binary or matchup product there is only one output data file. For each grid product there are actually two output files, one for ascending and the other for descending (orbits) observations. To simplify the flow diagrams, however, the grid outputs are represented by single boxes. The grids are generated at 0.5X2.0 degrees. For these grids, only the center CrIS FOV is matched and saved at a grid point. In each case the near center FOV is picked for the match. All validation products are direct access binary files. The details of these file formats are outlined in the NUCAPS ICD (NESDIS/STAR, 2008a).

The three upstream units are designed to process a single granule of data. The Validation unit, however, processes an entire day’s granules at once. Therefore, the unit only runs once a day. The unit script is triggered as a function of time, as opposed to the arrival or availability of data in the DHS. It runs on the previous day’s data and runs regardless of whether all the input data are available or not. Granules are often missing from a given day’s data so it does not make sense to wait for all data to be available. Furthermore, part of the purpose of the unit is to assess the timeliness of data being made available to the NUCAPS units.

## Directory Description

This section describes the directory structure of the source code (programs and scripts) and system file directories that are delivered to NDE in the Delivered Algorithm Package (DAP). The content and purpose of each subdirectory is briefly described. Note: this section does not describe the directory structure of the NUCAPS processing system. The directory structure for all NUCAPS processing is determined solely by the NDE system developers. The NUCAPS system itself does not assume any directory structure. The location (e.g. paths) of everything a NUCAPS unit requires during operation is passed to it through the NDE-generated PCF files. da website) this is on the S DRIVE??in the System Maintenance Manual Guideliness ????

Paths are always shown in italics. Because the system could be installed at any location on a given machine, the highest level directory path of installation for the system is always referred to as “*$BASE*”. Therefore, the location of all files and directories is described with respect to this high level directory. Since the NDE DHS will handle the staging of all NUCAPS files and directories, we only provide the source code and driver script description.

All driver scripts (Perl) for the four processing units are located at:

$BASE/OPS/scripts

Source code is only kept on the test machine. Only executable code is allowed on the production machine. All source code for those languages which are compiled (C/C++ and Fortran 77/90) is located at:

$BASE/SOURCE/

The preprocessor code for handling the GFS surface pressure information is at:

$BASE/SOURCE/gfs\_preprocessor

All the retrieval code is at:

$BASE/SOURCE/retrieval

All the regular NUCAPS system code is located at:

$BASE/SOURCE/code

All allocation subroutines and functions are located at:

$BASE/SOURCE/code/allocate

The Makefile configuration file provides information on file prefix and suffix structures for the compilation. It is located here:

$BASE/SOURCE/code/config

Code for the Digital Elevation Model is located here:

$BASE/SOURCE/code/dem

All Fortran 90 utilities, such as Planck functions, array manipulation functions, spherical distance averaging and spherical distance determination, are at:

$BASE/SOURCE/code/futils

All subroutines related to reading, writing, and mapping to the global grids are located here:

$BASE/SOURCE/code/global\_grids

All forecast file GRIB manipulation subroutines are located here:

$BASE/SOURCE/code/grib

All Fortran 90 HDF5 high-level readers and writers (specific to a given instrument) and utilities that handle low-level functions (like reading, writing, getting array dimensions) are located at:

$BASE/SOURCE/code/hdf5\_f90

All common modules such as system default integer sizes, common parameters, and error modules are located here:

$BASE/SOURCE/code/include

The level2 (retrieval) source code is located at:

$BASE/SOURCE/code/level2

All main programs (Fortran 90) and the system Makefile are located at:

$BASE/SOURCE/code/main

The code used for generating and writing the matchups are at:

$BASE/SOURCE/code/matchups

The modules for CrIS, ATMS, and NUCAPS\_ALL are at:

$BASE/SOURCE/code/modules

All NUCAPS system granule-slice monitoring code is at:

$BASE/SOURCE/code/monitoring

All Fortran 90 NetCDF high-level readers and writers (specific to a given instrument) are located at:

$BASE/SOURCE/code/netcdf\_readwrite\_f90

All Fortran 90 NetCDF utilities that handle low-level functions (like reading, writing, getting array dimensions) are at:

$BASE/SOURCE/code/netcdf\_utils\_f90

All code related to the generation of principal components and reconstructed radiances is located at:

$BASE/SOURCE/code/pc

All the Fortran 90 subroutine and modules for handling subset NUCAPS data, and reading/writing NUCAPS binary files, are located at:

$BASE/SOURCE/code/subset\_level1c\_products

All the C++ functions and modules for generating and manipulating the NUCAPS subsets are at:

$BASE/SOURCE/code/subset

All the system time functions (written in C) are located here:

$BASE/SOURCE/code/time

All the Fortran 90 subtroutines and modules for doing the VIIRS colocation are located at:

$BASE/SOURCE/code/viirs\_colocation

The NCEP Wesley code that is used for reading the GFS files is located here:

$BASE/SOURCE/code/Wesley

## Source Code Description

The NUCAPS system contains 666 programs, subprograms (functions and subroutines) and scripts written in Fortran 77/90, C/C++, and Perl. In total there are 173,238 lines of code. It is simply not possible to identify each program and describe its purpose in this section. Therefore, this section identifies the source code of each main program, notes its relative location in the source tree, and describes its function.

All of the main the programs located in *$BASE/SOURCE/code/main* subdirectory are Fortran 90 programs as indicated by having a *.f90* extension. When compiled, the Makefile for all these programs are configured to produce executables with the same name, but without the *.f90* extension. This makes their executable incarnations easily identifiable. For each main program, we supply the relative path and then a short description of its purpose follows.

$BASE/SOURCE/code/main/main\_nucaps\_preprocessor.f90

This program runs in the Preproc Unit. It is the main program to preprocess the NUCAPS HDF5 input data. This program reads 6 CrIS/ATMS HDF5 input data files for 3 granules, processes the data for the center granule, and writes the processed data to an output netCDF4 data file.

$BASE/SOURCE/code/main/main\_nucaps\_subset.f90

This program runs in the Subsetter Unit. It is the main program to subset NUCAPS data. This program creates the necessary input data structure, reads a NUCAPS\_ALL netCDF file, subsets, calculates principal components (optional), fills the output data structure, and writes the subset data to a NUCAPS subset netCDF data file.

$BASE/SOURCE/code/main/main\_gfs\_global\_grids.f90

This program can run in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in the GFS Grib Model Forecast data and the 0.5X2.0 satellite data grids. These data are then matched onto a single "forecast grid".

$BASE/SOURCE/code/main/main\_nucaps\_ccr\_to\_netcdf.f90

This program runs in the Retrieval Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a file called ccr.filenames that contains the name of the input CCR (Cloud Cleared Radiance) file and the output netCDF file. The routine reads the CCR file and writes the data out to the netCDF.

$BASE/SOURCE/code/main/main\_nucaps\_fullglobal\_grids.f90

This program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in NUCAPS netCDF subset granular files and a resource file called global\_grids.filenames to create a set of daily globally gridded files of day and nighttime data. The grid resolution is specified in the resource file. Unlike the standard global grids, this program writes all the available CrIS footprints to the global grid. Therefore, the input file list should contain the fuller spatial resolution netCDF files.

$BASE/SOURCE/code/main/main\_nucaps\_global\_binary.f90

The program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a file called nucaps\_binary.filenames that contains a list of the names of the CrIS level 1C subset netCDF files for 1 day. These files are read and their contents are placed in a direct access binary file. Each record consists of 1 CrIS FOV.

$BASE/SOURCE/code/main/main\_nucaps\_global\_grids.f90

The program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in the NUCAPS netCDF subset granular files and a resource file called global\_grids.filenames to create a set of daily globally grided files of day and nighttime data. The grid resolutions is specified in the resoure file.

$BASE/SOURCE/code/main/main\_nucaps\_global\_matchup\_binary.f90

This program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a file called nucaps\_binary.filenames that contains a list of the names of the NUCAPS level 1C subset netCDF files for 1 day. These files are read and their contents are placed in a direct access binary file. Each record consists of 1 CrIS FOV. However, the file is written so that 9 CrIS FOVs can be read at once. To do this the header records had to be enlarged by 9 times.

$BASE/SOURCE/code/main/ main\_nucaps\_global\_matchups.f90

This program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a resource file containing the names of files. A loop cycles through each of the radiosonde locations matching each one to the closest CrIS FOR (Field Of Regard) as it reads the global NUCAPS locational file. For each match, the record numbers from the NUCAPS locational file are stored for each radiosonde match. When the global binary file is read, only for those record numbers where there were matches are extracted and written out to the output binary. An output ASCII file contains only those sonde locations where NUCAPS matches were actually made. This output file is then passed on as an input to the ATMS matchups.

$BASE/SOURCE/code/main/ main\_nucaps\_granule\_slicer.f90

This program runs outside of NUCAPS processing in the OSPO QA environment. It reads in a file called nucaps\_slicer.filenames that contains the name of a single netCDF granule file to slice up. The data from the file are written out into a number of smaller files. One set of files contain only 1 channel for all CrIS FOVs. Another file set contains all the channels for a single FOV. The first set provides an image of the granule whereas the second set provides a spectrum at each observation point.

$BASE/SOURCE/code/main/ main\_nucaps\_l2\_global\_grids.f90

This is a main program that runs in the Validation Unit. It reads in the NUCAPS level2 retrieval netCDF granular files and a resource file called global\_grids.filenames to create a set of daily globally gridded files of day and nighttime data. The grid resolution is specified in the resource file (2x0.5).

$BASE/SOURCE/code/main/ main\_nucaps\_l2\_global\_matchup\_bin.f90

This program runs in the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a file called nucaps\_l2\_binary.filenames that contains a list of the names of the NUCAPS level 2 subset netCDF files for 1 day. These files are read and their contents are placed in a direct access binary file. Each record consists of 1 ATMS FOV.

$BASE/SOURCE/code/main/ main\_nucaps\_l2\_global\_matchups.f90

This program runs in the Validation Unit. It reads all the file record numbers in the NUCAPS EDR binary files and writes them out to a matchup record file. Therefore, the output matchup file contains matches with radiosondes. There may be missing EDR data, but there is still correspondence to the SDR record locations.

$BASE/SOURCE/code/main/main\_nucaps\_l2\_granule\_slicer.f90

This program runs outside of NUCAPS processing in the OSPO QA environment. It reads in a file called nucaps\_l2\_slicer.filenames that contains the name of a single netCDF granule file to slice up. The data from the file are written out into a number of smaller files. One set of files contain only 1 pressure level for all CrIS FORs. Another file set contains all the pressure levels data for a single FOR. The first set provides an image of the granule whereas the second set provides a profile at each observation point.

$BASE/SOURCE/code/main/main\_nucaps\_l2\_loc\_binary.f90

This program runs inside the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads in a file called nucaps\_l2\_binary.filenames that contains a list of the names of the NUCAPS level 2 netCDF files for 1 day. These files are read and their location contents (such as lat, lon, and time) are placed in a direct access binary file. Each record consists of 1 CrIS FOV.

$BASE/SOURCE/code/main/main\_nucaps\_l2\_to\_netcdf.f90

This program runs inside the Retrieval Unit. It reads in a file called l2.filenames that contains the name of the input EDR file and the output netCDF file. The program reads the EDR file and writes the data out to the netCDF. This reader is currently set up to read the EDR version GPV3B (iver = 19) files from Chris Barnet's retrieval. The records in the EDR file are written in sequential access and may vary in length. The netCDF file is dimensioned using the largest possible dimension for each record so all netCDF records are of equal length.

$BASE/SOURCE/code/main/main\_nucaps\_loc\_binary.f90

This program runs inside of the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. It reads a file called nucaps\_binary.filenames that contains a list of the names of the NUCAPS subset netCDF files for 1 day. It runs inside the Validation Unit. These files are read and their location contents (such as lat, lon, and time) are placed in a direct access binary file. Each record consists of 1 CrIS FOV.

$BASE/SOURCE/code/main/main\_nucaps\_netcdf\_to\_l2\_binary.f90

This program runs inside of the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. This program reads in a file called nucaps\_l2\_binary.filenames that contains the name of the output binary file and the name of the input NUCAPS level 1C subset netCDF file for 1 granule. This netCDF file is read and the data are written out to 2 sequential access binary files. This program is very similar to main\_nucaps\_global\_binary except there is only 1 input file and the output binary file is in Chris Barnet's input level 2 format. It runs inside the Retrieval Unit.

$BASE/SOURCE/code/main/main\_nucaps\_read\_matchups.f90

This program runs inside of the Validation Unit. It is currently deactivated for Phase 3 and will be removed in future releases. This program reads the SDR or EDR global matchups binary file, and writes the data records into an output text file. It runs inside the Validation Unit.

$BASE/SOURCE/code/main/main\_pc\_scores\_stats.f90

This program reads in the 3-Band PC file and computes statistical output for quality control of the sensors. It reads in a file called pc\_scores\_stats.filenames that contains the name of the 3-Band PC file as well as the output filename. It runs inside the Subset Unit.

$BASE/SOURCE/code/main/main\_cris\_geo2txt.f90

This program reads in the CrIS GEO file and converts it to the text format required for the CrIS-VIIRS collocation code. It reads in a file called cris\_geo2text.filenames that contains the names of the hdf5 binary dump files for the variables needed, as well as the names of the output text files for the data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_viirs\_cf2bin.f90

This program reads in the VIIRS Cloud Fraction file and converts it to a binary file required for the CrIS-VIIRS collocation code. It reads in a file called viirs\_cf2bin.filenames that contains the names of the hdf5 binary dump files for the variables needed, as well as the name of the output test files for the data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_viirs\_cfgeo2txt.f90

This program reads in the VIIRS Cloud Fraction GEO file and converts it to a text file required for the CrIS-VIIRS collocation code. It reads in a file called viirs\_cfgeo2txt.filenames that contains the names of the hdf5 binary dump files for the variables needed, as well as the name of the output test files for the data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_viirs\_ch2bin.f90

This program reads in the VIIRS Cloud Height file and converts it to a binary file required for the CrIS-VIIRS collocation code. It reads in a file called viirs\_ch2bin.filenames that contains the names of the hdf5 binary dump files for the variables needed, as well as the name of the output test files for the data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_viirs\_chgeo2bintxt.f90

This program reads in the VIIRS Cloud Height GEO file and converts it to a binary and text file required for the CrIS-VIIRS collocation code. It reads in a file called viirs\_chgeo2bintxt.filenames that contains the names of the hdf5 binary dump files for the variables needed, as well as the name of the output test files for the data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_cris\_viirs\_cf\_colocation.f90

This program reads in the CrIS binary file created by main\_cris\_geo2txt and the VIIRS Cloud Fraction files created by main\_viirs\_cf2bin and viirs\_cfgeo2txt then collocates the data and writes the output to the NUCAPS\_ALL netCDF file. It reads in a file called main\_cris\_viirs\_cf\_colocation.filenames that contains the NUCAPS ALL netCDF file, ancillary look up tables locations, as well as the binary and text files of the CrIS and VIIRS data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_cris\_viirs\_ch\_colocation.f90

This program reads in the CrIS binary file created by main\_cris\_geo2txt and the VIIRS Cloud Height files created by main\_viirs\_ch2bin and viirs\_chgeo2bintxt then collocates the data and writes the output to the NUCAPS\_ALL netCDF file. It reads in a file called main\_cris\_viirs\_ch\_colocation.filenames that contains the NUCAPS ALL netCDF file, ancillary look up tables locations, as well as the binary and text files of the CrIS and VIIRS data. It runs inside the Preprocessor Unit.

$BASE/SOURCE/code/main/ main\_cris\_olr.f90

This program runs the Outgoing Longwave Radiance code. It reads in a file called olr.filenames that contains the NUCAPS\_ALL netcdf file, the ancillary files needed as well as all the required netCDF metadata information. It runs inside the Preprocessor Unit.

*$BASE/SOURCE/*src\_util/airsb.F

This is the main program that runs the entire NUCAPS retrieval code. It runs inside the Retrieval Unit. The executable version of this program is called airsb103.exe. This program requires two input arguments: a resource file (nucaps\_retrieval.filenames) and a text string identifying the year month day and granule number being processed. When this executable is invoked it assumes that it has the seven namelist files it required to be located in the current working directory. These namelist provide all the required input parameters to run. These files are identified in the Appendix and greater detail about their content is provided in the NUCAPS ATBD.

GFS preprocessor code is here:

$BASE/SOURCE/gfs\_preprocessor/ preprocessor.cpp

This is the C++ source for converting the GFS binary files produced by wgrib2 to the binary input format required by the retrieval. The executable form of this program is called preprocessor.exe. The program runs inside the Retrieval unit. A pair of GFS GRIB2 files whose forecast coverage times bound the CrIS granule are converted to a set of flat binary format files by the wgrib2 tool. Then, preprocessor.exe reads these files and the associated NUCAPS full resolution file and outputs a pair of identical files containing surface pressure interpolated temporally and spatially to the CrIS FOVs.

# NORMAL OPERATIONS

## System Control

### System Control Files

This section describes the input controls required to operate NUCAPS. Before any one of the four NUCAPS unit driver scripts is invoked by the NDE system, the NDE system creates a working directory and a Process Control File (PCF) specific to that unit and to that given run. The PCF is placed in the working directory and then an instance of the driver script is run. The driver script, in turn, assumes the PCF is located in the current working directory. The driver script is then run by the NDE system. When the driver script finishes, in addition to its output products, it produces a Process Status File (PSF), and returns a value to the calling NDE process. The NDE system then uses this to obtain information to determine whether the run was success and which files are available for distribution or transfer to the next stage of processing.

### Processing Controls

There are four PCF and four PSF files in NUCAPS. These files are associated with the operation of each NUCAPS unit (Preproc, Subset, Retrieval, and Validation). The PCF file will contain the names to the files to be processed along with their full path (location in the file system). The PSF just contains the names of the output files that were generated successfully. A description of the various fields of each unit’s PCF and PSF are provided below followed by an example.

NUCAPS\_Preproc.pl.PCF

job\_coverage\_start – This is the start time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

job\_coverage\_end – This is the end time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

SATELLITE\_ID – This should be set to npp, but is ready for when we switch to the the J01 satellite.

OPS\_BIN – This is the path to the location where all the NUCAPS program executables are stored. These are the compiled Fortran programs. Perl scripts are located elsewhere.

SCRIPT\_OPS – This is the path to the location where all the NUCAPS Perl scripts are stored.

UUIDGEN – The location of the UUID generation Linux command.

NUCAPS\_ALL\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS all netCDF file.

CRIS\_SDR – This is the name and location (if not in the local working directory) of the CrIS SDR HDF5 target granule file.

CRIS\_GEO – This is the name and location (if not in the local working directory) of the associated CrIS SDR GEO HDF5 target granule file that matches the CRIS\_SDR.

ATMS\_TDR\_1 – This is the name and location (if not in the local working directory) of the ATMS TDR HDF5 file whose time precedes that of the target granule.

ATMS\_TDR\_2 – This is the name and location (if not in the local working directory) of the ATMS TDR HDF5 target granule file.

ATMS\_TDR\_3 – This is the name and location (if not in the local working directory) of the ATMS TDR HDF5 file whose time follows that of the target granule.

ATMS\_GEO\_TARGET\_1 – This is the name and location (if not in the local working directory) of the ATMS TDR GEO HDF5 file whose time precedes that of the target granule.

ATMS\_GEO\_TARGET\_2 – This is the name and location (if not in the local working directory) of the ATMS TDR GEO HDF5 target granule file.

ATMS\_GEO\_TARGET\_3 – This is the name and location (if not in the local working directory) of the ATMS TDR GEO HDF5 file whose time follows that of the target granule.

VIIRS\_PROCESSING\_FLAG – This is a flag to activate the VIIRS cloud collocation with CrIS. This will allow output of VIIRS data into the CrIS SDR BUFR downstream.

VIIRS\_COUNT – This indicates the number of VIIRS granules that overlay the current CrIS granule (this ranges from 1 to 2).

VIIRS\_CF\_1 – This is the VIIRS cloud mask file.

VIIRS\_CF\_GEO\_1 – This is the VIIRS cloud mask geolocation file.

VIIRS\_CH\_1 – This is the VIIRS cloud top product file.

VIIRS\_CH\_GEO\_1 – This is the VIIRS cloud top product geolocation file.

VIIRS\_ALL\_TEMPLATE – This is the VIIRS netCDF4 template file.

VIIRS\_COLOCATION\_LUT\_DIR – This is the VIIRS collocation lookup table.

APOD – This is the apodization flag (either “yes” or “no”). Nominally this is yes (on), but we allow the option for this to be turned off.

DEM\_FILE – This is the name and location (if not in the local working directory) of Global Digital Elevation Model (DEM) file (Global\_DEM.bin). It’s probably best to specify this path to this file or link it to the local working directory since it is very large.

DEM\_LAT\_PTS – This is the latitude resolution of the DEM file (currently 21600).

DEM\_LON\_PTS – This is the longitude resolution of the DEM file (currently 43200).

RESAMPLE\_DATA – This is the name and location (if not in the local working directory) of the ATMS footprint resampling coefficient file. This is a static file.

H5DUMP\_LOC – This is the location of the h5dump command (part of the HDF5 library utilities).

CRIS\_STATUS\_CHECK\_FLAG – This is the CrIS status check flag. This flag allows for checking of the operational status of the instrument in the headers of the CrIS HDF5 files. If the instrument is not operational and this flag is “ON” to allow checking, the CrIS data will not be processed.

ATMS\_STATUS\_CHECK\_FLAG – This is the ATMS status check flag. This flag allows for checking of the operational status of the instrument in the headers of the ATMS HDF5 files. If the instrument is not operational and this flag is “ON” to allow checking, the ATMS data will not be processed. In this case, CrIS data can still be processed by the Subsetter Unit, but the retrieval unit should not run.

OLR\_PROCESSING\_FLAG – This flag activates the generation of the CrIS OLR product.

OLR\_TEMPLATE – This is the netCDF4 template file for the CrIS OLR product.

OLR\_DIR – This is a path to a directory where OLR static files are stored.

OLR\_COR – This is the OLR radiance correction regression coefficient file.

OLR\_REG – This is the OLR regression coefficient file.

Below is an example of a NUCAPS\_Preproc.pl.PCF file.

job\_coverage\_start=201504292011059

job\_coverage\_end=201504292011357

SATELLITE\_ID=npp

OPS\_BIN=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/Common\_Bin

SCRIPT\_OPS=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/scripts

UUIDGEN\_LOC=/usr/bin/uuidgen

NUCAPS\_ALL\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_all.nc

CRIS\_SDR=SCRIS\_npp\_d20150429\_t2011059\_e2011357\_b18152\_c20150430023041151664\_noaa\_ops.h5

CRIS\_GEO=GCRSO\_npp\_d20150429\_t2011059\_e2011357\_b18152\_c20150430022355251268\_noaa\_ops.h5

ATMS\_TDR\_1=TATMS\_npp\_d20150429\_t2010313\_e2011029\_b18152\_c20150430023039460578\_noaa\_ops.h5

ATMS\_TDR\_2=TATMS\_npp\_d20150429\_t2011033\_e2011349\_b18152\_c20150430023039460578\_noaa\_ops.h5

ATMS\_TDR\_3=TATMS\_npp\_d20150429\_t2011353\_e2012069\_b18152\_c20150430023039460578\_noaa\_ops.h5

ATMS\_GEO\_TARGET\_1=GATMO\_npp\_d20150429\_t2010313\_e2011029\_b18152\_c20150430022338779252\_noaa\_ops.h5

ATMS\_GEO\_TARGET\_2=GATMO\_npp\_d20150429\_t2011033\_e2011349\_b18152\_c20150430022338779252\_noaa\_ops.h5

ATMS\_GEO\_TARGET\_3=GATMO\_npp\_d20150429\_t2011353\_e2012069\_b18152\_c20150430022338779252\_noaa\_ops.h5

VIIRS\_PROCESSING\_FLAG=ON

VIIRS\_COUNT=1

VIIRS\_CF\_1=IICMO\_npp\_d20150429\_t2010221\_e2011445\_b18152\_c20150430025147546875\_noaa\_ops.h5

VIIRS\_CF\_GEO\_1=GMODO\_npp\_d20150429\_t2010221\_e2011445\_b18152\_c20150430024536021943\_noaa\_ops.h5

VIIRS\_CH\_1=VCTHO\_npp\_d20150429\_t2010221\_e2011445\_b18152\_c20150430025146804871\_noaa\_ops.h5

VIIRS\_CH\_GEO\_1=GCLDO\_npp\_d20150429\_t2010221\_e2011445\_b18152\_c20150430024535556208\_noaa\_ops.h5

VIIRS\_ALL\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/viirs\_all\_1gran.nc

VIIRS\_COLOCATION\_LUT\_DIR=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/viirs\_colocation

APOD=yes

DEM\_FILE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/DEM/Global\_DEM.bin

DEM\_LAT\_PTS=21600

DEM\_LON\_PTS=43200

RESAMPLE\_DATA=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/resample/resample.data

H5DUMP\_LOC=/opt/apps/ots/hdf5/hdf5-1.8.7/bin/h5dump

CRIS\_STATUS\_CHECK\_FLAG=OFF

ATMS\_STATUS\_CHECK\_FLAG=OFF

OLR\_PROCESSING\_FLAG=ON

OLR\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/cris\_olr.nc

OLR\_DIR=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/olr

OLR\_COR=rad\_corr\_reg\_coef\_17boxcar\_airsv10ab\_2.asc

OLR\_REG=olr\_reg\_coef\_cv005\_17boxcar\_2.asc

NUCAPS\_Preproc.pl.PSF

The NUCAPS Preproc PSF file contains three fields. The first is the name of the NUCAPS all-channel, all FOV output file. It shouldn’t need a path since it should be produced in the local working directory. Below is an example. The second file is an internal metadata file required by the retrieval unit. The third is the OLR netCDF output file. Note that there is no field identifier tag (followed by an equal sign) in the PSF files, only the name of the output file itself.

NUCAPS 1317-CrIS channel file containing all FOVs

NUCAPS Bounding Box internal metadata file

CrIS OLR granule file

Below is an example.

NUCAPS\_ALL\_20150429\_2011059\_2011357.nc

Bounding\_Box\_20150429\_2011059\_2011357.txt

NUCAPS-OLR\_v1r0\_npp\_s201504292011059\_e201504292011357\_c201509021922140.nc

NUCAPS\_Subset.pl.PCF

job\_coverage\_start – This is the start time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

job\_coverage\_end – This is the end time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

SATELLITE\_ID – This should be set to npp, but is ready for when we switch to the the J01 satellite.

OPS\_BIN – This is the path to the location where all the NUCAPS program executables are stored. These are the compiled Fortran programs. Perl scripts are located elsewhere.

NUCAPS\_C0300\_ALLFOVS\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel subset file for all CrIS FOVs.

NUCAPS\_C0300\_CENTERFOV\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel subset file for only the center CrIS FOV (out of each 3X3 CrIS FOR).

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel all FOVs principal component (PCS) 1-band (1B) for all FOVs.

NUCAPS\_C0300\_ALLFOVS\_PCS3B\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel all FOVs principal component (PCS) 3-band (3B) for all FOVs.

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel center FOV principal component (PCS) 1-band (1B) for all FOVs.

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel center FOV principal component (PCS) 3-band (3B) for the center FOVs.

NUCAPS\_C1317\_1SCAN\_TEMPLATE – This is the name and location (if not in the local working directory) to the template for the NUCAPS 300-CrIS channel principal component (PCS) 3-band (3B) for the center FOVs.

PC\_1BAND\_COEFF – This is the name and location (if not in the local working directory) of the eigenvector file for CrIS band 1.

PC\_2BAND\_COEFF – This is the name and location (if not in the local working directory) of the eigenvector file for CrIS band 2.

PC\_3BAND\_COEFF – This is the name and location (if not in the local working directory) of the eigenvector file for CrIS band 3.

PC\_FBAND\_COEFF – This is the name and location (if not in the local working directory) of the eigenvector file for all CrIS bands.

NUCAPS\_ALL\_PRODUCT – This is the name and location (if not in the local working directory) of the NUCAPS all input file (produced upstream by NUCAPS\_Preproc.pl).

NUCAPS\_C0300\_ALLFOVS\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel subset file for all CrIS FOVs.

NUCAPS\_C0300\_CENTERFOV\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel subset file for only the center CrIS FOV (out of each 3X3 CrIS FOR).

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel principal component (PCS) 1-band (1B) for all FOVs.

NUCAPS\_C0300\_ALLFOVS\_PCS3B\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel principal component (PCS) 3-band (3B) for all FOVs.

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel principal component (PCS) 1-band (1B) for the center FOVs.

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 300-CrIS channel principal component (PCS) 3-band (3B) for the center FOVs.

NUCAPS\_C1317\_1SCAN\_FLAG – This is a flag (“ON” or “OFF”) to activate the production of the NUCAPS 1317-CrIS channel file containing only all FOVs from 1 scan (thinned from the input granule).

Below is an example of a NUCAPS\_Subset.pl.PCF file:

job\_coverage\_start=201504292011059

job\_coverage\_end=201504292011357

SATELLITE\_ID=npp

OPS\_BIN=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/Common\_Bin

NUCAPS\_C0300\_ALLFOVS\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_allfovs.nc

NUCAPS\_C0300\_CENTERFOV\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_centerfov.nc

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_allfovs\_pcs1b.nc

NUCAPS\_C0300\_ALLFOVS\_PCS3B\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_allfovs\_pcs3b.nc

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_centerfov\_pcs1b.nc

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c0300\_centerfov\_pcs3b.nc

NUCAPS\_C1317\_1SCAN\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_c1317\_1scan.nc

PC\_1BAND\_COEFF=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/pc\_coeffs/eigvec\_20120515\_band1\_ascii\_real

PC\_2BAND\_COEFF=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/pc\_coeffs/eigvec\_20120515\_band2\_ascii\_real

PC\_3BAND\_COEFF=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/pc\_coeffs/eigvec\_20120515\_band3\_ascii\_real

PC\_FBAND\_COEFF=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/pc\_coeffs/eigvec\_20120515\_full\_ascii\_real

NUCAPS\_ALL\_PRODUCT=NUCAPS\_ALL\_20150429\_2011059\_2011357.nc

NUCAPS\_C0300\_ALLFOVS\_FLAG=ON

NUCAPS\_C0300\_CENTERFOV\_FLAG=ON

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_FLAG=ON

NUCAPS\_C0300\_ALLFOVS\_PCS3B\_FLAG=ON

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_FLAG=ON

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_FLAG=ON

NUCAPS\_C1317\_1SCAN\_FLAG=ON

NUCAPS\_Subset.pl.PSF

The NUCAPS Subset PSF file contains the names of all seven output files. These should all be located in the local working directory. The files are as follows:

NUCAPS 399-CrIS channel subset file for all CrIS FOVs.

NUCAPS 399-CrIS channel subset file for only the center CrIS FOV (out of each 3X3 CrIS FOR).

NUCAPS 1317-CrIS channel file containing only all FOVs from 1 scan (thinned from the input granule).

NUCAPS 399-CrIS channel principal component (PCS) 1-band (1B) for all FOVs.

NUCAPS 399-CrIS channel principal component (PCS) 1-band (1B) for the center FOVs.

NUCAPS 399-CrIS channel principal component (PCS) 3-bands (3B) for the center FOVs.

NUCAPS PCS monitoring file.

Below is an example.

NUCAPS\_C0300\_ALLFOVS\_20150429\_2011059\_2011357.nc

NUCAPS\_C0300\_CENTERFOV\_20150429\_2011059\_2011357.nc

NUCAPS\_C1317\_1SCAN\_20150429\_2011059\_2011357.nc

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_20150429\_2011059\_2011357.nc

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_20150429\_2011059\_2011357.nc

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_20150429\_2011059\_2011357.nc

NUCAPS-PCS-MONITORING\_v1r0\_npp\_s201504292011059\_e201504292011357\_c201508061047340.txt

NUCAPS\_Retrieval.pl.PCF

job\_coverage\_start – This is the start time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

job\_coverage\_end – This is the end time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

SATELLITE\_ID – This should be set to npp, but is ready for when we switch to the the J01 satellite.

SCRIPT\_OPS – This is the path to the location where all the NUCAPS Perl scripts are stored.

OPS\_BIN – This is the path to the location where all the NUCAPS program executables are stored. These are the compiled Fortran programs. Perl scripts are located elsewhere.

NUCAPS\_ALL\_PRODUCT – This is the name and location (if not in the local working directory) of the NUCAPS all input file (produced upstream by NUCAPS\_Preproc.pl).

BOUNDING\_BOX\_PRODUCT – This is the name and location (if not in the local working directory) of the NUCAPS internal metadata file which contains the bounding box and ascending/descending node status.

WGRIB\_LOC – This is the name and location of the wgrib2 executable command (currently in /opt/apps/ots/bin/wgrib2).

WGRIB\_COMMAND – This is the name and location of the NUCAPS wgrib2 preprocessing Perl script (run\_wgrib2.pl)

CC\_DAY\_FILENAME – This is the name and location (if not in the local working directory) of the clear flag day time file needed by the retrieval.

CC\_NIGHT\_FILENAME – This is the name and location (if not in the local working directory) of the clear flag night time file needed by the retrieval.

AVN\_FILE1 – This is the name and location (if not in the local working directory) of the first GFS GRIB2-format forecast file needed by the retrieval.

AVN\_FILE2 – This is the name and location (if not in the local working directory) of the second GFS GRIB2-format forecast file needed by the retrieval.

LAC\_FILENAME – This is the name and location (if not in the local working directory) of the Local Angle Correction (LAC) file name used by the retrieval. If none is specified the LAC will not be applied.

NUCAPS\_L2\_TEMPLATE – This is the name and location (if not in the local working directory) of the NUCAPS EDR template netCDF file for the profile data.

NUCAPS\_AWIPS\_TEMPLATE – This is the name and location of the AWIPS netCDF4 template file.

NUCAPS\_CCR\_TEMPLATE – This is the name and location (if not in the local working directory) of the NUCAPS CCR template netCDF file for the Cloud-Clear Radiances (CCR)

NUCAPS\_CCR\_AR\_TEMPLATE – This is the name and location (if not in the local working directory) of the NUCAPS CCR file for the CLASS archive.

H5DUMP\_LOC – The full path of the h5dump command (a utility in the HDF5 library).

UUIDGEN\_LOC – The full path of the uuid\_gen command. Note, this will be a different path and even a different command name in Linux.

AWIPS\_FLAG – This should normally be “OFF” for running in NDE. This allows the AWIPS subset to be produced in the NUCAPS test environment at STAR.

BYTESWAP\_FLAG – This is a flag to invoke byte swapping in the retrieval preprocessor. This may be required for running on Linux if everything was compiled as big endian.

TUNINGCOEFFILE – This is the retrieval tuning coefficient file.

TUNINGMASKFILE –This is the retrieval tuning mask file.

NOAAEIGFILE – This is the NOAA IR regression radiance eigenvector file.

NOAAFRQFILE – This is the NOAA IR regression frequency file.

NOAAREGFILE – This is the NOAA IR regression coefficient file.

NOAAANGFILE – This is the NOAA IR regression angle coefficient file.

L2ERROR\_IN – This is a file containing the ensemble error estimate of climatology.

OLRCOEFFILE – This is contains the rapid transmittance coefficients to compute outgoing longwave

HSBWEIGHTFILE – This is a microwave weighting file.

ECOFFILE – This is a microwave retrieval error covariance file.

MWCOVFILE – This is a microwave retrieval covariance file.

UARSCLIMFILE – This is the UARS climatology file for upper atmosphere.

NCEPCLIMFILE – This is the NCEP climatology file for Temperature and water vapor.

RTAERRFILE – This is the radiative transfer model error file.

SOLARFILE – This is the CrIS solar irradiance file for the radiance calculation.

TC\_AIRS – This is the post-flight CrIS RTA coefficient file.

IRNOISEFILE – This is post-flight CrIS RTA coefficients file.

MASUDAFILE – This is coefficients file for the Masuda surface emissivity model for ocean.

MWNOISEFILE – This is microwave noise file.

TC\_AMSU – This is the ATMS transmittance coefficient file.

CLDAVGFILE – This is the cloud averaging table.

IO\_NL – This is the input/output namelist file.

CLOUDS\_NL – This the clouds namelist file.

MICROW\_NL – This is the microwave namelist file.

OZONE\_NL – This is the trace gas namelist file.

PRO\_NL – This is the profile namelist file.

TEMP\_NL – This is the temperature namelist file.

WATER\_NL – This is the water vapor namelist file.

Below is an example.

job\_coverage\_start=201504292011059

job\_coverage\_end=201504292011357

SATELLITE\_ID=npp

SCRIPT\_OPS=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/scripts

OPS\_BIN=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/Common\_Bin

NUCAPS\_ALL\_PRODUCT=NUCAPS\_ALL\_20150429\_2011059\_2011357.nc

BOUNDING\_BOX\_PRODUCT=Bounding\_Box\_20150429\_2011059\_2011357.txt

WGRIB\_LOC=/opt/apps/ots/wgrib2/wgrib2

WGRIB\_COMMAND=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/scripts/run\_wgrib2.pl

CC\_DAY\_FILENAME=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/retrieval/regress/clearflag/L2.I.cleartest\_coef.v2.0.2.day.anc

CC\_NIGHT\_FILENAME=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/retrieval/regress/clearflag/L2.I.cleartest\_coef.v2.0.2.night.anc

AVN\_FILE1=gfs.t12z.pgrb2.1p00.f006

AVN\_FILE2=gfs.t12z.pgrb2.1p00.f009

LAC\_FILENAME=none

NUCAPS\_L2\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_l2.nc

NUCAPS\_AWIPS\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_awips.nc

NUCAPS\_CCR\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_ccr.nc

NUCAPS\_CCR\_AR\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_ccr\_archive.nc

H5DUMP\_LOC=/opt/apps/ots/hdf5/hdf5-1.8.7/bin/h5dump

UUIDGEN\_LOC=/usr/bin/uuidgen

AWIPS\_FLAG=OFF

BYTESWAP\_FLAG=ON

TUNINGCOEFFILE=cris\_20120515\_v10a\_fin.asc

TUNINGMASKFILE=120214\_cris\_tuning\_mask.asc

NOAAEIGFILE=cris\_eigvec\_05122015

NOAAFRQFILE=cris\_v03.frq

NOAAREGFILE=cris\_ccr\_05142015.asc

NOAAANGFILE=cris\_cld\_05122015.asc

L2ERROR\_IN=jpl\_100.inp

OLRCOEFFILE=airs\_olr.dat

HSBWEIGHTFILE=L2.M.weight.hsb.v1.0.0.anc

ECOFFILE=L2.M.ecof\_705.v1.0.0.anc

MWCOVFILE=L2.M.cov100av.v1.0.0.anc

UARSCLIMFILE=L2.uars\_clim.v1.0.3.anc

NCEPCLIMFILE=ncep\_clim.bin

RTAERRFILE=cris\_rtaerr\_v10a.asc

SOLARFILE=cris\_solar\_v10a.txt

TC\_AIRS=binary.trcoef.cris.v10a

IRNOISEFILE=tobin120120.dat

MASUDAFILE=L2.masuda.v2.0.0.anc

MWNOISEFILE=atms\_1.dat

TC\_AMSU=tr\_atms\_new.dat

CLDAVGFILE=cris\_v10a.t1

IO\_NL=io\_cris.nl

CLOUDS\_NL=clouds\_cris.nl

MICROW\_NL=microw\_cris.nl

OZONE\_NL=ozone\_cris.nl

PRO\_NL=pro\_cris.nl

TEMP\_NL=temp\_cris.nl

WATER\_NL=water\_cris.nl

NUCAPS\_Retrieval.pl.PSF

The NUCAPS Retrieval PSF file contains the names of all four output files. These should all be located in the local working directory. The files are as follows:

NUCAPS EDR file containing the trace gases and other profiles.

NUCAPS EDR monitoring file.

NUCAPS CCR file containing the Cloud-Clear Radiances (CCR). This contains ATMS data as well.

NUCAPS CCR file for the CLASS archive. This does not contain ATMS data, only CrIS CCR data.

Below is an example.

NUCAPS-EDR\_v1r0\_npp\_s201504292011059\_e201504292011357\_c201508061450500.nc

NUCAPS-EDR-MONITORING\_v1r0\_npp\_s201504292011059\_e201504292011357\_c201508061450500.txt

NUCAPS\_CCR\_20150429\_2011059\_2011357.nc

NUCAPS-CCR-AR\_v1r0\_npp\_s201504292011059\_e201504292011357\_c201508061450500.nc

NUCAPS\_Validation.pl.PCF

Note that all the validation products except the EDR and OLR grids are deactivated for Phase 3. They will be removed for Phase 4.

job\_coverage\_start – This is the start time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

job\_coverage\_end – This is the end time of the granule in YYYYMMDDHHMMSSS format. NUCAPS needs this to create the time strings for its output file names and to perform some internal time checks.

SATELLITE\_ID – This should be set to npp, but is ready for when we switch to the the J01 satellite.

OPS\_BIN – This is the path to the location where all the NUCAPS program executables are stored. These are the compiled Fortran programs. Perl scripts are located elsewhere.

SCRIPT\_OPS – This is the location of all the NUCAPS Perl scripts.

NUCAPS\_L2\_TEMPLATE – This is the netCDF4 template file used for the EDR.

MATCHUPS\_TABS – This is the name and location (if not in the local working directory) of the radiosonde matchup template file.

WGRIB\_LOC – This is the name and location of the wgrib2 executable command (currently in /opt/apps/ots/bin/wgrib2).

AVN\_GFS\_FILE – This entry is repeated five times. It contains the names of the 00 hour (analysis) GFS files. The files are listed in this order 00Z, 06Z, 12Z, and 18Z for the day – 1 and for 00Z of the current day. These are needed for the GFS global grids.

NUCAPS\_C1317\_1SCAN\_BINARY\_FLAG – The flag to activate the 1-scan SDR binary.

NUCAPS\_FG\_GRIDS\_FLAG – The flag to activate the 3X3 SDR global grids.

NUCAPS\_GG\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 SDR global grids.

NUCAPS\_FG\_PCS1B\_GRIDS\_FLAG – The flag to activate the 3X3 PCS 1-band global grids.

NUCAPS\_GG\_PCS1B\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 PCS 1-band global grids.

NUCAPS\_GG\_PCS3B\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 PCS 3-band global grids.

NUCAPS\_GG\_GFS\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 GFS global grids.

NUCAPS\_GG\_L2\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 EDR global grids.

NUCAPS\_GG\_CCR\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 CCR global grids.

NUCAPS\_OLR\_GRIDS\_FLAG – The flag to activate the 0.5X2.0 OLR global grids.

NUCAPS\_SDR\_MATCHUPS\_FLAG – The flag to activate the SDR global matchups.

NUCAPS\_L2\_MATCHUPS\_FLAG – The flag to activate the EDR global matchups.

NUCAPS\_C1317\_1SCAN\_PRODUCT – This is the name of the input NUCAPS 1317-CrIS channel file containing only all FOVs from 1 scan (thinned from the input granule). This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_GG\_GRIDS\_PRODUCT – This is the name of the input NUCAPS 300-CrIS channel subset file for only the center CrIS FOV (out of each 3X3 CrIS FOR). This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_FG\_PCS1B\_GRIDS\_PRODUCT – This is the name of the input NUCAPS 300-CrIS channel principal component (PCS) 1-band (1B) for all FOVs. This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_GG\_PCS1B\_GRIDS\_PRODUCT – This is the name of the input NUCAPS 300-CrIS channel principal component (PCS) 1-band (1B) for the center FOVs. This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_GG\_PCS3B\_GRIDS\_PRODUCT – This is the name of the input NUCAPS 300-CrIS channel principal component (PCS) 3-band (3B) for the center FOVs. This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_SDR\_GLOBAL – This is the name the input “NUCAPS all” files containing all CrIS channels and all FOVs. This entry is repeated 3375 times (nominally) for all the granules required for 1 day’s worth of processing. These data are used for both the SDR global matchups and the 3X3 SDR global grids. This should consist of 30 hours of files. For example, if the target day to be processed is November 27th, this list should contain the last 3 hours of files from November 26th (21Z-00Z), all the files from November 27th, and the first 3 hours of files from November 28th (00Z-03Z).

NUCAPS\_EDR\_GLOBAL – This is name of the input CrIS EDR files. These EDR files contain the trace gases and other geophysical profiles. This entry is repeated 3375 times (nominally) for all the granules required for 1 day’s worth of processing. These data are used for both the EDR global matchups and the 3X3 EDR global grids. This should consist of 30 hours of files. For example, if the target day to be processed is November 27th, this list should contain the last 3 hours of files from November 26th (21Z-00Z), all the files from November 27th, and the first 3 hours of files from November 28th (00Z-03Z).

NUCAPS\_CCR\_PRODUCT – This is name of the input CrIS CCR files. NUCAPS CCR file containing the Cloud-Clear Radiances (CCR). This contains ATMS data as well. This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

NUCAPS\_OLR\_PRODUCT – This is the name of the input CrIS OLR files used for the OLR global grids. This entry is repeated 2700 times (nominally) for all the granules required for 1 day’s worth of processing.

Below is only a partial example due to the length of this file.

job\_coverage\_start=20110222000000

job\_coverage\_end=20110223000000

job\_coverage\_start=20150428000000

job\_coverage\_end=20150429000000

SATELLITE\_ID=npp

OPS\_BIN=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/Common\_Bin

SCRIPT\_OPS=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/scripts

NUCAPS\_L2\_TEMPLATE=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/CDLFILES/nucaps\_l2.nc

MATCHUPS\_TABS=/utilraid/data/users/tking/NUCAPS/DAP\_20150504/NUCAPS/CrISOPS/matchups/radiosonde\_matchup\_template

WGRIB\_LOC=/opt/apps/ots/wgrib2/wgrib2

AVN\_GFS\_FILE=gfs.t00z.pgrb2.1p00.f000.20150428

AVN\_GFS\_FILE=gfs.t06z.pgrb2.1p00.f000.20150428

AVN\_GFS\_FILE=gfs.t12z.pgrb2.1p00.f000.20150428

AVN\_GFS\_FILE=gfs.t18z.pgrb2.1p00.f000.20150428

AVN\_GFS\_FILE=gfs.t00z.pgrb2.1p00.f000.20150429

NUCAPS\_C1317\_1SCAN\_BINARY\_FLAG=OFF

NUCAPS\_FG\_GRIDS\_FLAG=OFF

NUCAPS\_GG\_GRIDS\_FLAG=OFF

NUCAPS\_FG\_PCS1B\_GRIDS\_FLAG=OFF

NUCAPS\_GG\_PCS1B\_GRIDS\_FLAG=OFF

NUCAPS\_GG\_PCS3B\_GRIDS\_FLAG=OFF

NUCAPS\_GG\_GFS\_GRIDS\_FLAG=OFF

NUCAPS\_GG\_L2\_GRIDS\_FLAG=ON

NUCAPS\_GG\_CCR\_GRIDS\_FLAG=OFF

NUCAPS\_OLR\_GRIDS\_FLAG=ON

NUCAPS\_SDR\_MATCHUPS\_FLAG=OFF

NUCAPS\_L2\_MATCHUPS\_FLAG=OFF

NUCAPS\_C1317\_1SCAN\_PRODUCT=NUCAPS\_C1317\_1SCAN\_20150428\_0000099\_0000397.nc.

.

NUCAPS\_GG\_GRIDS\_PRODUCT=NUCAPS\_C0300\_CENTERFOV\_20150428\_0000099\_0000397.nc

.

.

NUCAPS\_FG\_PCS1B\_GRIDS\_PRODUCT=NUCAPS\_C0300\_ALLFOVS\_PCS1B\_20150428\_0000099\_0000397.nc

.

.

NUCAPS\_GG\_PCS1B\_GRIDS\_PRODUCT=NUCAPS\_C0300\_CENTERFOV\_PCS1B\_20150428\_0000099\_0000397.nc

.

.

NUCAPS\_GG\_PCS3B\_GRIDS\_PRODUCT=NUCAPS\_C0300\_CENTERFOV\_PCS3B\_20150428\_0000099\_0000397.nc

.

.

NUCAPS\_SDR\_GLOBAL=NUCAPS\_ALL\_20150428\_0000099\_0000397.nc

.

.

NUCAPS\_EDR\_GLOBAL=NUCAPS-EDR\_v1r0\_npp\_s201504280000099\_e201504280000397\_c201504281350050.nc

.

.

NUCAPS\_CCR\_PRODUCT=NUCAPS\_CCR\_20150428\_0000099\_0000397.nc

.

.

.

NUCAPS\_OLR\_PRODUCT=NUCAPS-OLR\_v1r0\_npp\_s201504280000099\_e201504280000397\_c201504281116510.nc

NUCAPS\_Validation.pl.PSF

NUCAPS global-coverage binary file containing 1 scan of full spatial and spectral resolution CrIS and ATMS radiance and brightness temperatures data from every 6th granule.

NUCAPS 3X3 degree global-coverage grid file containing CrIS and ATMS radiances and brightness temperatures obtained from ascending orbits only.

NUCAPS 3X3 degree global-coverage grid file containing CrIS and ATMS radiances and brightness temperatures obtained from descending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS and ATMS radiances and brightness temperatures obtained from ascending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS and ATMS radiances and brightness temperatures obtained from descending orbits only.

NUCAPS 3X3 degree global-coverage grid file containing CrIS principal components derived from all 3 CrIS bands combined and ATMS brightness temperatures obtained from ascending orbits only.

NUCAPS 3X3 degree global-coverage grid file containing CrIS principal components derived from all 3 CrIS bands combined and ATMS brightness temperatures obtained from descending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS principal components derived from all 3 CrIS bands combined and ATMS brightness temperatures obtained from ascending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS principal components derived from all 3 CrIS bands combined and ATMS brightness temperatures obtained from descending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS principal components derived from each of the 3 CrIS bands separately and ATMS brightness temperatures obtained from ascending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS principal components derived from each of the 3 CrIS bands separately and ATMS brightness temperatures obtained from descending orbits only.

NUCAPS radiosonde matchup file containing all the CrIS and ATMS radiance and brightness temperature data at each CrIS FOR for the nearest collocation to a radiosonde launch location.

NUCAPS radiosonde location text file containing the list of known launch locations at which there were collocations with CrIS observations.

NUCAPS radiosonde matchup file containing all the CrIS/ATMS retrieval (profile) data at each CrIS FOR for the nearest collocation to a radiosonde launch location.

NUCAPS 3X3 degree global-coverage grid file containing GFS forecast data at the locations of CrIS and ATMS radiance and brightness temperatures obtained from ascending orbits only.

NUCAPS 3X3 degree global-coverage grid file containing GFS forecast data at the locations of CrIS and ATMS radiance and brightness temperatures obtained from descending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS and ATMS retrieval (profiles) data obtained from ascending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing CrIS and ATMS retrieval (profiles) data obtained from descending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing cloud-cleared CrIS radiances and ATMS brightness temperatures obtained from ascending orbits only.

NUCAPS 0.5X2 degree global-coverage grid file containing cloud-cleared CrIS radiances and ATMS brightness temperatures obtained from descending orbits only.

Below is an example. Note that all the validation products except the EDR and OLR grids are deactivated for Phase 3. They will be removed for Phase 4.

NUCAPS-GG-EDR-GRIDS-ASC\_v1r0\_npp\_s20150428000000\_e20150429000000\_c20150831114639.bin

NUCAPS-GG-EDR-GRIDS-DSC\_v1r0\_npp\_s20150428000000\_e20150429000000\_c20150831114639.bin

NUCAPS-GG-OLR-GRIDS-ASC\_v1r0\_npp\_s20150428000000\_e20150429000000\_c20150831114639.bin

NUCAPS-GG-OLR-GRIDS-DSC\_v1r0\_npp\_s20150428000000\_e20150429000000\_c20150831114639.bin

## Installation

Setup of NUCAPS occurs in four stages: (1) unpacking the DAP, (2) compiling of the code, (3) installation of the compiled code and system files into the NDE production system, (4) and configuration of the NDE system using the production rules supplied in the NUCAPS documentation. This section identifies the items to be installed

### Setup Requirements

This section lists all the items that are delivered as part of the NUCAPS DAP that are to be used for generating the NUCAPS products. The NUCAPS DAP produces four subdirectories when unpacked, two of these are the *OPS* and *SOURCE* subdirectories.

All source code is located in the SOURCE subdirectory, the contents of which are document in section 3.2:

$BASE/SOURCE/

NUCAPS system code is here:

$BASE/SOURCE/code

And the Make for the system code is here:

$BASE/SOURCE/code/main/Makefile

GFS preprocessor code is here:

$BASE/SOURCE/gfs\_preprocessor

And the Makefile for all the GFS code is:

$BASE/SOURCE/gfs\_preprocessor/Makefile.gfs\_preprocessor

The NUCAPS retrieval code is here:

$BASE/SOURCE/retrieval

And the Makefile for all the retrieval code is here:

$BASE/SOURCE/retrieval/rebuild.com

All of the NUCAPS system files are here in the OPS directory:

$BASE/OPS/

This is the location of the Digital Elevation Model (DEM):

$BASE/OPS/DEM/ Global\_DEM.bin

This is the location of the principal components eigenvector coefficients files:

$BASE/OPS/pc\_coeffs/

This is the location of the radiosonde matchup template files:

$BASE/OPS/matchups radiosonde\_matchup\_template

This is the location of the ATMS footprint resample coefficient file:

$BASE/OPS/resample/resample.data

This subdirectory contains a directory tree of all the retrieval LUTs and coefficient files that are described later in Table 7‑6 in the Appendix:

$BASE/OPS/retrieval

This subdirectory contains the static ancillary LUTs need for the CrIS-VIIRS collocation code:

$BASE/OPS/viirs\_colocation

This subdirectory contains all the NUCAPS Perl driver scripts and utilities:

$BASE/OPS/scripts

The four driver scripts that the NDE system will run are located here. They are (1) NUCAPS\_Preproc.pl, (2) NUCAPS\_Subset.pl, (3) NUCAPS\_Retrieval.pl, and (4) NUCAPS\_Validation.pl.

### Installation Procedures

The NDE Data Handling System (DHS) “registers” all necessary information for executing instances of NOAA Unique Product (NUP) algorithms in the Oracle Database. Algorithms are executed when all necessary input data has arrived and met all defined criteria for execution. Production jobs are created in the database with all necessary information for that particular instance of execution and are placed in a processing queue. A processing node picks up the queued job and a Process Control File (PCF) is created with all the necessary information for the production job in the database. At this time a driver script for executing the algorithm is called on the appropriate processing node. Therefore, installation of an algorithm into NDE is identical for all algorithms. The first step is to build the executables on the NDE Configuration Management (CM) build machines followed by deployment of those executables (and all other necessary data, e.g. scripts) to the processing nodes. After this, it is simply a matter of registering (to the database) various XMLs describing the algorithm and how to execute it. The following is the minimum set of XMLs required:

* Product Definition XML(s): Each product has its own product definition file which describes all identifying information for a product such as product short name, file name pattern, product quality summary attributes, size, data retention period, etc. This enables the NDE system to ingest products and make them available for product generation or distribution. All ingested input and output products for an algorithm must have a Product Definition XML.
* Algorithm Definition XML: Each algorithm has a single algorithm definition file describing the general characteristics of an algorithm such as the algorithm name and version, the location of the executables, the name of the driver script, names of all input/output products (only those that require ingest - no static ancillary data needs to be identified), and any algorithm parameter names.
* Production Rule Definition XML: Each algorithm may have one or more production rules associated with it. The production rule file describes how to execute an instance of an algorithm such as specific input data and its characteristics, output data, algorithm parameter values (e.g. flags, location of static data), execution characteristics (temporal refresh, gazetteer, timeout thresholds waiting for input, etc.)

After all XMLs associated with an algorithm and it’s production rules have been written, these are then registered to the database using registration scripts:

* registerProduct.pl
* registerAlgorithm.pl
* registerProductionRule.pl
* registerNodeAlgorithm.pl - associates an algorithm with a particular processing node or nodes which allows the node to pick the produciton job up off the queue

Depending on the production rule, a Gazetteer XML may need to be registered. This just describes a region of the Earth that a production rule should be executed in. If this is the case, then registerGazetteer.pl should be run before the registerProductionRule.pl.

Once the hardware, connectivity, and software discussed in section 4.2.1are installed, the NUCAPS System Software can be installed.

The Phase 3 DAP will consist of one tar’d and gzip’d file:

NUCAPS\_v3-0\_20150504.tar.gz

This file name adheres to the NDE DAP file name convention described in section 4.1 of Standards for Algorithm Delivery and Integration Using Delivered Algorithm Packages (DAPs), Version 1.3 (NDE, 2011).

Unpacking the system package in the *$BASE* directory to install the system:

gunzip NUCAPS\_v3-0\_20150504.tar.gz

tar xvf NUCAPS\_v3-0\_20150504.tar.gz

This will create 4 subdirectories in the current working directory.

SOURCE/ - location of all source code (e.g. Fortran 90/77 and C/C++ code).

OPS/ - location of all scripts and required system files (static files like LUTs, PCS coefficient files, the DEM)

TEST/ - location of sample data for testing the DAP

DOCS/ - location of all delivered documents (principally the SMM, EUM, TRD).

The first step is to edit the *$BASE/SOURCE/code/main/Makefile* fileto add the correct paths of the netCDF4, HDF5, and SZLIB (NETCDFINC, NETCDFLIB, HDF5INC, HDF5LIB, SZLIB). And if you want to use the command `make install` to easily copy over all of the executables to another directory you can edit the path for INSTALL\_DIR.

Change into the main directory where *Makefile* is located.

cd $BASE/SOURCE/code/main/

On SADIE these libraries are currently located at:

NETCDFINC=/opt/apps/ots/netcdf/netcdf-4.1.1/include

NETCDFLIB=/opt/apps/ots/netcdf/netcdf-4.1.1/lib

HDF5INC=/opt/apps/ots/hdf5/hdf5-1.8.5-patch1/include

HDF5LIB=/opt/apps/ots/hdf5/hdf5-1.8.5-patch1/lib

SZLIB=/opt/apps/ots/szip/szip-2.1/lib

Note that SZLIB is defined a couple places in the Makefile.

Note, if the code is later recompiled with different libraries, the netCDF4 template files must be recreated using ncgen from the new netCDF4 library. For reference, the netCDF4 template files are located here:

$BASE/OPS/CDLFILES

To compile the source code:

make clean

make

make install

The `make install` command will copy all of the executables to the directory specified as the INSTALL\_DIR in the make file.

The GFS and retrieval Makefiles do not require any modification since they don’t rely on any outside libraries.

For the GFS preprocessor code do the following:

cd $BASE/SOURCE /gfs\_preprocessor

make -f Makefile.gfs\_preprocessor clean

make -f Makefile.gfs\_preprocessor

cp preprocessor.exe $DESTINATION

For the retrieval code do the following:

cd $BASE/SOURCE /retrieval

rebuild.com gfortran

cp exe/airsb103.exe $DESTINATION

## Build

### Compilation Procedures

All NOAA Unique Product (NUP) algorithms are compiled on the Redhat Enterprise Linux (RHEL) operating system statically linked to the standard set of GNU Compiler Collection (GCC) compilers.

NUP science algorithms are compiled on dedicated build machines under Configuration Management (CM) control.

### Build Procedure

NDE Algorithm Integrators generate a set of build instructions (or in some cases scripts) for CM control and builds. These instructions are specific to each Delivered Algorithm Package (DAP) and are created using the Dev environment for unit testing. The CM Lead builds the algorithms on the appropriate build machine following the controlled instructions (and/or script). The algorithm binaries and all necessary driver scripts, internal static data, tables, etc. are deployed across all processing nodes in the Test Environment. All source code is maintained by the Subversion CM tool and is not deployed to the Test or Production Environments. After verification of the algorithm in the Test Environment, it is promoted to the Production Environment. The same binaries and associated scripts and data are deployed to the /opt/data/nde/NDE\_OP1/algorithms/<ALGORITHM NAME> folder on each processing node.

## Operations Procedures

### Normal Operations

The NDE Data Handling System (DHS) consists of Ingest (INS), Product Generation (PGS), Product Distribution (PDS), and Monitoring subsystems, and is physically spread across numerous servers, both IBM and Linux based, and utilizes a Storage Area Network (SAN) for all data retention. NDE DHS uses the open source JBoss Service-Oriented Architecture (SOA) platform Java-based middleware to provide the infrastructure (messaging, clustering, failover, load balancing, security, etc) for the INS, PGS, PDS, and Monitoring subsystems. These subsystems are further decomposed into applications with each application built and managed independently. Applications are built using the Enterprise Service Bus (ESB) framework provided by JBoss. NDE DHS consists of five ESB applications which perform the core tasks for ingest, product generation and product distribution.  These five applications are Ingest, PGSFactory, DISTFactory,  PGSProcessor, and DISTProcessor and are summarized below. Each JBoss server logs all messages, warnings, and errors in persistent log files that are refreshed every 24 hours.

The DHS utilizes the Oracle Spatial 11g Database  Management System for all back end data storage and geospatial support.  All persistent data, whether it is the catalog of products, the definition and registration of software in the NDE framework, or the messages passed between NDE services, are managed by the Oracle database. DHS access to the backend Oracle database is accomplished with the use of stored procedures, version controlled software (queries) precompiled for efficiency.

In the following paragraphs, each subsystem is described as data flows through the system. Figure 0‑2 illustrates the data flow through each subsystem.

Ingest Subsystem (INS)

Ingest (INS) is responsible for the receipt and catalog of all NDE related files. After successful validation, incoming product files (Raw/Sensor/Environmental Data Records or xDRs, NOAA Unique Products or NUPs, intermediate products, and ancillary data) and their identifying information (i.e. file metadata) are registered to the database as they arrive in the landing zone of the SAN. Files are processed by the INS in a Last-In First-Out (LIFO) fashion allowing the most current data to be ingested first if a backlog occurs. This is accomplished using the ingest throttle (a Perl script) which sorts valid files by observation time and makes them available to the INS application.

Each arriving file has an associated product alert file (PAF) which  initiates the ingest throttle. The PAFs for xDRs distributed by the Joint Polar Satellite System (JPSS) Integrated Data Processing Segment (IDPS) system are Cyclic Redundancy Check (CRC-32) files generated by IDPS in order to monitor file integrity. All other incoming product files (NUPs, intermediate products, and ancillary data) have NDE generated COMP files which are simply text files containing the associated filename.

The INS servers also handle the acquisition of all ancillary data required by NDE science algorithms from the Environmental Satellite Processing Center (ESPC) Data Distribution Server (DDS) via a File Transfer Protocol (FTP pull) service. The FTP downloader service monitors pull directories for new files (files that have not already been downloaded) via a loop. When new files are made available, the downloader ftp pulls them onto the SAN landing zone and caches them to prevent duplicate pulls.

When a new product file and associated PAF arrives in the SAN landing zone (incoming\_input directory) and the ingest throttle has performed it’s LIFO function, the following occurs (the software implements these as services with messaging down the service pipeline):

1. Validate incoming file against the registered products in the database.
   * Check PAF file name against registered products and their file name patterns in the database, they must match exactly one pattern.
   * If a the pattern matching fails then the file is moved to the ingest\_failed directory on the SAN, an error is logged in the INS server log file, and continued messaging through the services ceases.
2. Execute integrity checks on incoming product files.
   * Retrieves the ingest processing steps defined for each product in the product description table of the database. These steps include a metadata extractor routine (e.g. h5dump, ncdump, or custom extractor scripts) and may include a crc checksum (e.g. with IDPS xDRs). At a minimum, the metadata extractor must be able to retrieve the file observation date, and start/end time. Additional information is retrieved from IDPS xDRs and NUPs such as geolocation of a granule, day/night flag, quality summaries, etc.
   * Executes any defined checksums (CRC-32 or MD5). If checksums fail, the file is moved to the ingest\_failed directory on the SAN, an error is logged in the INS server log file,  and continued messaging through the services ceases.
3. Persist the product file metadata in the database.
   * Execute the defined metadata extractor routine.
   * Insert extracted metadata into the database. If a file has already been ingested (i.e. file metadata has been inserted in the database), it is rejected as a duplicate, moved to the ingest\_failed directory on the SAN, an error is logged in the INS server log file,  and continued messaging through the services ceases.
   * Move file to the products directory on the SAN.
   * Move PAF file to the incoming\_processed directory on the SAN.
   * Send message to PGS Factory and DIST Factory that a new file has arrived.

NDE DHS utilizes two Linux nodes running JBoss servers for INS. Both ingest servers are JBoss clustered allowing for workload sharing between the ingest application nodes. If either INS node is unavailable, the other node will take over all of the processing.

Product Generation Subsystem (PGS)

NDE PGS handles all data management and processing for NUP science algorithms and product file tailoring for xDRs and NUPs. File tailoring includes aggregation of granules, reformatting, subsampling, and file thinning (selection of specific arrays/variables). PGS implements two overarching applications: the factory (PGSFactory) which manages all potential production jobs and the processor (PGSProcessor) which executes production jobs.

NUP science algorithms and product tailoring production is defined by a set of definition files which describe the algorithm and its specific execution criteria. There is one algorithm definition file for each algorithm. This file describes the general algorithm characteristics such as the algorithm name, driver script name, version, input/output data (product short names), and algorithm parameter names (e.g. flags, directory names). Each algorithm may have one or more production rule definition files. The production rule definition file describes the specific execution criteria for running an algorithm such as temporal and geospatial characteristics (e.g. refresh rate and geographical regions for execution), input data characteristics (e.g. imagery solar bands for day granules only), algorithm parameter values (e.g. flag values, paths to static ancillary directories), and output product names. The definition files are controlled and are used to register the information into the database via Perl scripts.

PGS Factory

All production rules identify a “TRIGGER” input product. When a trigger product file is ingested, a message is sent from INS to the PGSFactory initiating the PGS process which begins with the creation of a production job spec. The following occurs and is managed by the PGSFactory (the software implements these as services with messaging down the service pipeline):

1. Create a production job spec.
   * Factory queries the database for all production rules that have defined the ingested product as a trigger file.
   * Creates a production job spec for all production rules with the defined trigger product.
2. Monitor production job specs for completion via a loop. A production job spec is not complete until all the required input data has been ingested and covers the observation temporal period while meeting all production rule criteria. A production job spec can time out, i.e., if required input data does not arrive in a predetermined period of time (defined in the production rule). The clock for timing out begins when the production job spec is created.
   * Retrieve all input data and associated file metadata for the completed (or timed-out) production job spec. Product files that have been defined as required input for the production rule and that meet the temporal criteria will be retrieved.
   * Filter out any input data that does not pass specific tests. For example, a production rule may define an input product as valid only if the observation was during the day (contained in a day/night flag metadata). If a product file does not pass a test, then the product file is not considered valid for the production job spec.
   * Evaluate input file accumulation threshold. A minimum threshold for temporal coverage can be defined for an input product. If the retrieved input data that has passed the filter tests does not cover enough of the production job spec observation time period, then the production job spec is completed but has a status of “COMPLETE-NOINPUT” and no production job is created.
3. Complete the production job spec after all input data has been ingested, passed the filter tests, and met the minimum file accumulation threshold.
4. Create a production job and queue it for the PGSProcessor.

NDE DHS utilizes two Linux nodes running JBoss servers for the factories. Both factory servers are JBoss clustered allowing for workload sharing between the factory application nodes. If either factory node is unavailable, the other node will take over. It should be noted that the factories are not resource intensive so these servers support both the production and distribution factories and also stage the NDE DHS Portal (GUI interface).

PGS Processor

Algorithms (including NDE tailoring) are assigned to dedicated processing nodes which make up the bulk of NDE DHS servers. There are AIX and Linux-based servers (i.e. nodes) that algorithms can be assigned to (mapped to in the database) depending on what platform they are compiled on. Processing nodes are independent (not clustered) allowing for the simple addition of new servers if capacity is reached.

Each processing node runs a JBoss server for the PGSProcessor application which handles the execution of queued production jobs assigned to it. Each node is limited with the number of production jobs that can be run simultaneously by the database setting for production job boxes allowing for load balancing. The setting is configurable and can be production rule specific.

After a processing node has picked a production job off the database queue (via looping), the following occurs and is managed by the PGSProcessor (the software implements these as services with messaging down the service pipeline):

1. Copy input files to working directory.
   * Creates working directory on processing node local RAID.
   * Copies all required input files for the production job from the SAN product directory to the working directory.
   * Create the Process Status File (PSF) in the working directory. This file is empty until the algorithm returns from execution and writes all output file names to it.
   * Create the Process Control File (PCF). A Process Control File (PCF) is created in the working directory which lists (name=value) all necessary information for the production job (e.g. input data file names, production rule parameters, working directory location, job id). This file has all the information an algorithm requires for executing a particular production job.
2. Run the production job. The algorithm driver script is called by the JBoss server and is given the working directory path as an argument. This allows the algorithm driver script to read the PCF and call all other scripts and binaries for normal execution.
   * Algorithm returns with a code of zero (success) and writes the output filenames to the PSF.
   * Algorithm returns with a non-zero code (failure).
3. Perform error handling.
   * After an algorithm returns and regardless of return code, PGSProcessor scans the algorithm log file for any error/warning messages and enters them in the database.
   * A non-zero return code from the algorithm causes the PGSProcessor to compress the entire working directory into a zip file and moves it to the forensics directory on the local RAID of the processing node.
4. Get output filenames written to the PSF. Validate file name patterns. All output product files must be registered to the database.
5. Copy output files to the SAN landing zone for ingest. The PGSProcessor reads the PSF file written by the algorithm for filenames. These files are copied from the local working directory to the SAN landing zone for ingest along with generated PAF files.
6. Release the job box for another queued job to be picked up by the processing node.

Product Distribution Subsystem (PDS)

NDE PDS manages all product distribution to customers via subscriptions. All customers are registered in the database and have subscriptions for each desired product. NDE implements File Transfer Protocol Secure (FTPS) push/pull for distribution. Similar to PGS, the PDS implements two applications: DISFactory which creates and manages all distribution jobs and the DISProcessor which executes all push distribution jobs.

NDE DHS utilizes two Linux nodes running JBoss servers for the factories. Both factory servers are JBoss clustered allowing for workload sharing between the factory application nodes. If either factory node is unavailable, the the other node will take over. It should be noted that the factories are not resource intensive so these servers support both the production and distribution factories and also stage the NDE DHS Portal (GUI interface).

There are four dedicated Linux-based servers for both FTPS push and pull jobs for a total of eight distribution platforms. The pull servers interface with a WebMux which load balances across the four pull servers. The WebMux is where pull customers interface with NDE DHS. The pull servers monitor the pull log and update the database when a customer has pulled a particular file. The four push servers run the Jboss server for the DISProcessor application. The eight servers are located in a different security zone (DMZ) than the rest of the NDE DHS since they must interface with external customer sites.

DIS Factory

After a product file has been ingested, the INS servers send a message to the DISFactory that a subscribed product has arrived. The following occurs and is managed by the DISFactory (the software implements these as services with messaging down the service pipeline):

1. Qualify the product file for distribution.
   * Create a distribution prepare request (DPR) for the product file and distribution requests (DRs) for all subscriptions associated with the product. A DPR is similar to the PGS production job spec; it contains all the information required for a distribution job. A single DPR can service several customer subscriptions if they are subscribed to the same product. There is only one DR for each customer subscription mapping to the DPR for that product (one DPR to many DRs is possible).
   * Check the data file against any predefined tests set in the subscription. A customer may specify in the subscription that a product data file must have particular file metadata met such as day/night flags, orbit number, file size, ascending/descending indicator, etc. A customer may also filter data based on quality using the global product quality summaries for IDPS and NUP products. If a file fails any filter tests, then the DR for that subscription is removed from the database. If all DRs for a DPR are removed then the DPR is also removed.
2. Retrieve DPRs for completion in a LIFO fashion.
3. Compress product files if the subscription specifies this. Outgoing product files can be compressed using gzip or zip compression. Compressed files are created on the SAN in the dist folder.
4. Generate a checksum if the subscription specifies this. Outgoing product files can have a checksum file created using CRC-32 or MD5. Checksums are created on the SAN in the dist folder.
5. Create pull links to product files. If the DPR specifies a pull subscription, then the DISFactory creates a link to the product file in the dist directory on the SAN.
6. Create and queue a distribution job and complete the distribution prepare request and associated distribution requests.
7. Create user links for pull subscriptions. Customers that have pull subscriptions pull from a link in their user folder on the SAN which points to the link created in the dist folder which, in turn, points to the product file in the product directory. At this point, pull distribution jobs are ready for the customer and no further action is taken by PDS until the file has been pulled.
8. Create notification request if pull subscription specifies this. There are a couple of options available to customers for notification such as email and SOAP messages. The request is queued for the DISProcessor to execute.

DIS Processor

The NDE DHS DIS Processor handles the execution of all push distribution jobs and notifications. Similar to the PGSProcessor, the DISProcessor JBoss server runs on each push distribution node. There are distribution job boxes assigned to each node for load balancing push distributions. After a distribution node has picked a push distribution job off the database queue (via looping), the following occurs and is managed by the DISProcessor (the software implements these as services with messaging down the service pipeline):

1. Execute FTPS push of the distribution job product file to the customer.
2. Create and queue a notification request if push subscription specifies this. There are a couple of options available to customers for notification such as email and SOAP messages.
3. Execute notification requests.

Release the distribution job box for another queued job to be picked up by the distribution node.

### Data Preparation

The primary source of input data for NDE is the Joint Polar Satellite System (JPSS) Interface Data Processing Segment (IDPS). The IDPS maintains a connection to the NDE SAN and writes all Suomi-National Polar Partnership (NPP) data files or xDRs (Sensor Data Records, Environmental Data Records, and some Raw Data Records) to the NDE incoming\_input directory. In addition, IDPS generates and writes accompanying checksum files to NDE ingest. The checksums are generated using CRC-32 polynomial division. The CRC files are used by the NDE Ingest subsystem to check the integrity of the IDPS xDR file (after NDE generates it’s own checksum for comparison). If a file fails the checksum comparison, it is not ingested into NDE and is placed in an ingest\_failed directory and noted in the ingest log. During ingest, NDE reads particular metadata fields from the xDR files, which are in Hierarchical Data Format version 5 (HDF5), using the HDF5 library h5dump executable. If this fails, the product is moved to the ingest\_failed directory and is noted in the ingest log.

The other source for external data in the NDE system is the Environmental Satellite Processing and Distribution (ESPDS) Data Distribution System (DDS) which provides all necessary algorithm ancillary data. The NDE Ingest subsystem File Transfer Protocol (FTP) pulls from the DDS servers. There is no checksum capability available for DDS to NDE data. All FTP pull traffic is monitored by the FTP pull log scanner which logs any errors or warnings. NDE extracts the necessary file metadata (file observation start/end times) from the file name using internally developed scripts. If this extraction fails, the file is moved to the ingest\_failed directory and is noted in the ingest log.

This is the extent of NDE interaction with an ingested file (besides copying files for execution). All other I/O with ingested files is performed by the algorithm itself. If an algorithm is not able to read a file (e.g. file is corrupted), the algorithm is expected to exit with an appropriate error message in the algorithm log and return a non-zero code to NDE. This will cause the NDE processing node to compress the working directory into a forensics file which is moved to a forensics folder for later analysis.

## Distribution

### Data Transfer / Communications

Distribution methods are either a push or pull transaction via FTP Secure (FTPS) protocol. FTPS allows for authentication encryption but no encryption of the data itself. This is more efficient than sFTP protocol which encrypts both. There are eight (8) distribution servers, four (4) for push customers and four (4) for pull customers allowing for high availability of distribution. The NDE Distribution subsystem offers notification options for customers. They include email, Simple Object Access Protocol (SOAP) message, or a file-based Data Availability Notification (DAN).

### Distribution Restrictions

There is no restriction regarding the release of data products to users, however, only real time users will be served from OSPO distribution system, the non real-time users may order the data from the CLASS.

### Product Retention Requirements

The data will be retained for 96 hours on NDE operation system. The Quality Monitoring system will retain the level-1 data for 4 days and level-2 data for 10 days in order to generate the statistics. Image data will be kept for longer period of time for generating the time series and pattern reorganization studies. The disk space requirement for NUCAPS QA system is two (2) TB. This server will have similar capability as Sandia. The Quality Assurance Tools will be deployed on the server which are basically modified tools developed for IASI data monitoring and data QA.

### External Product Tools

No external product tools are supplied. The NUCAPS output files are plain text files, binary files, or netCDF4 files. External users can choose their own tools to display and analyze these output files.

# MONITORING AND MAINTENANCE

## Job Monitoring

Monitoring of the status of the job will be performed by the OSPO operators on a 24 X 7 basis. The monitoring procedures for product generation will be provided by the NDE system developers. The products will be monitored every 30 minutes to ensure their uninterrupted production. The product monitoring also includes the ancillary data as inputs to the products generation.

During the day time the product monitoring function and data quality assurance are also performed by the OSPO contractor and the sounding PAL using the IASI graphical user interface available on Sandia. This interface works as intranet and requires a “Process Status” text file from the NUCAPS production server. The NUCAPS operational server will push the process status file to the NUCAPS QA server for products monitoring and data quality assurance.

## NUCAPS Quality Assurance

NUCAPS Quality Assurance (QA) will be performed by the OSPO contractor programmers and the sounding PAL using the QA tools. The QA tools developed for IASI will be modified for NUCAPS data Quality Monitoring. This will require a dedicated hardware similar to Sandia to host the QM tools and data. The data quality assurance is a post processing function, the data granules are aggregated and statistics are generated to produce plots and images. The images are also used for animation and time series studies to evaluate the systematic and uncorrelated errors. The statistics are generated include the daily, weekly, monthly and yearly products. NUCAPS yield (percentage of number of good retrievals over the globe) is monitored to show the usefulness of hyperspectral data. Several geophysical quantities such as temperature, water vapor, cloud clear radiance, trace gases, cloud parameters, and ozone will be monitored on a daily basis.

## Product Monitoring

### Unit Test Plans

The test plans and test results are documented in the NUCAPS Test Readiness Document located in Google Docs, and can be accessed by requesting it from the PAL.

### Internal Product Tools

Two programs are supplied for offline use at OSPO for quality monitoring purposes. These programs are referred to as “slicers” such that they input granules of NUCAPS SDR or EDR data and output small binary files containing (1) image data of the channels or geophysical properties like temperature, moisture and trace gasses and (2) spectra or profiles at each FOV. The granule is therefore sliced vertically and horizontally like cake to produce small files of data that can then be used by visualization tools in a quality monitoring environment.

The two slicer programs are *main\_nucaps\_granule\_slicer* and *main\_nucaps\_l2\_granule\_slicer* are provided to monitor the product outputs.

*main\_nucaps\_granule\_slicer* reads in a file called nucaps\_slicer.filenames that contains the name of a single netCDF granule file to slice up. The data from the file are written out into a number of smaller files. One set of files contain only 1 channel for all CrIS FOVs. So, if there are 1317 channels in the input file, there would be 1317 files. Another file set contains all the channels for a single FOV. So, if there are 120 CrIS FORs and 9 CrIS FOVs within each CrIS FOR, there would be 1080 (=120\*9) files. The first set of files can be thought of as "image" files. The second set of files are "spectrum" files. The first set provides an image of the granule whereas the second set provides a spectrum at each observation point.

*main\_nucaps\_l2\_granule\_slicer* reads in a file called nucaps\_l2\_slicer.filenames that contains the name of a single netCDF granule file to slice up. The data from the file are written out into a number of smaller files. One set of files contain only 1 pressure level for all CrIS FORs. So, if there are 100 pressure levels in the input file, there would be (8\*100=800) files (100 files each for Temperature, H2O\_Layer\_Col\_Den, O3\_Layer\_Col\_Den, Liquid\_H2O\_Layer\_Col\_Den, Ice\_Liquid\_Flag, CO\_Layer\_Col\_Den, CH4\_Layer\_Col\_Den and CO2\_Dry\_MixRatio). Another file set contains all the pressure levels data for a single FOR. So, if there are 120 CrIS FORs in a granule, there would be 960 (=120\*8) files. The first set of files can be thought of as "image" files. The second set of files are "profile" files. The first set provides an image of the granule whereas the second set provides a profile at each observation point.

### Performance Statistics

IASI QA tools reside on Sandia and use about 800 GB disk space. Image generation consists of two modes, the first mode is called static mode which generates a set of fixed number of images every day for a set of parameters, the second mode so called dynamic mode generates the images based on demand. A sub set of images generated by static mode are also available on the web server for public consumption. The dynamic mode is only used for deep dive data analysis when data anomalies are observed.

NUCAPS requires a similar hardware and software like IASI QA system. OSPO contractor programmers and the sounding PAL will modify the code written in Perl and Java for IASI for NUCAPS data monitoring. Some tools may not work and require to be developed and deployed on this sever that are specific to NUCAPS data. The statistics performance is performed on the basis of daily, weekly, monthly, and yearly. Statistics for percentage of number of good retrievals over the globe, known as yield will be monitored for NUCAPS to show the usefulness of an hyperspectral instrument. Several geophysical quantities such as temperature, water vapor, cloud clear radiance, trace gases, cloud parameters, and ozone will be monitored on a daily basis and will produce the weekly, monthly, and yearly products.

The two slicer programs described in section 5.3.2.are delivered as part of the DAP, but are intended for use by the PAL’s development team at OSPO. OSPO developers incorporate these two programs into a QA system they build and operate on hardware separate from NDE or the OSPO production environment. OSPO has developed a QA visualization system for IASI that is anticipated to be extended for NUCAPS. This system includes a QA website on which the output files of the slicer code are plotted. Due to the huge volume of information made available by the slicer (~16.5 million files/day), these plots are only meant to be a place to look in the event of a problem detected elsewhere in the system. These plots provide a way to view individual granules or the details of spectra and geophysical profiles at a very high resolution. This site resides behind the NSOF firewall and is therefore only viewable by the OSPO developers and sounding PAL as well as those certain STAR developers who have access to the NSOF VPN.

These slicer programs slice the NUCAPS SDR and EDR files into millions of small files. For the NUCAPS SDR, there is a file for the spectrum on each CrIS FOV and there is a file for the image of each channel from each granule.

1080 CrIS FOVs/granule \* 2700 granules/day = 2,916,000 spectra files/day

1305 CrIS channels/FOV \* 2700 granules/day = 3,523,500 image files/day

Therefore, there are 6,439,500 sliced SDR files per day.

For the EDR, there are the 100-level profiles for each of the 17 geophysical variables (O3, temperature, H2O, etc) on each CrIS FOR and then there is a file for each image file of the17 geophysical fields on each granule.

17 variables \* 120 CrIS FORs/granule \* 2700 granules/day = 5,508,000 profile files/day

17 variables \* 100 levels \* 2700 granules/day = 4,590,000 image files/day

Therefore, there are 10,098,000 EDR files/day.

The SDR slicer uses the NUCAPS “ALL” netCDF4 file as input:

NUCAPS\_ALL\_${YYYYMMDD}\_${HHMMSSS}\_${HHMMSSS}.nc

Where:

*${YYYYMMDD}\_${HHMMSSS}\_${HHMMSSS}* – A string of 3 date strings where:

YYYYMMDD – the year/month/day

HHMMSSS – the first string is the starting hour/minute/second

HHMMSSS – the second string is the ending hour/minute/second

The EDR slicers uses the NUCAPS EDR netCDF4 file as output:

NUCAPS-EDR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc

This file follows the NDE naming convention identified in the NDE document entitled Standards for Algorithm Delivery and Integration Using Delivered Algorithm Packages (DAPs) Version 1.3 (NDE, 2011).

### Product Monitoring

The monitoring is performed using several methods, currently IASI system send email notifications twice a day using the system logs to show the operational system status, image visualizing method is used for data products quality checks and data gaps analysis, two and three dimensional plots, scatter plots, and histogram are used for data quality monitoring. The image products are produced for granule, daily products, weekly products, monthly products, and yearly products. Clean up scripts are employed automatically to clean the data older than 4 days after generating the higher level products except in some cases the binary data are kept a little longer for dynamic page generation.

### Product Criticality

NUCAPS is identified as a mission critical and therefore a 24 X 7 service maintenance level is required. Return to service within 2 hours is required. .

## Maintenance

### Monitoring

The NDE Data Handling System (DHS) provides a Graphical User Interface (GUI) for monitoring all of the subsystem functionality (ingest, product generation, and distribution), resource utilization (CPUs, memory, storage), and system performance. The GUI is provided through the interactive NDE DHS Portal and consists of a Dashboard monitoring overall system health and a series of pages and links for increasingly detailed looks at the various subsystems. For further details on monitoring, see the NDE Operations Handbook and the NDE Software User’s Manual. The following summarizes simple ways to monitor ingest, product generation, and product distribution.

Monitoring begins with the NDE DHS dashboard where the high level summary of the state of the system is described. The window shows the state of all subsystems and their associated nodes, i.e. servers, (green = nominal, red = down). All windows show the backlog and throughput for each subsystem. The ingest window shows the number of failed ingests over the user defined period (e.g. 2 hours in this screen shot). The production window summarizes the number of failed production jobs over the defined time period (Last XX hrs. button), and the excessive run time jobs – these are jobs that have been running far too long, i.e., stuck in a processing state. Finally, the distribution window shows the number of failed and expired (did not meet customer defined latency) distribution jobs. All of the summaries are clickable and will take the user to a more detailed page.

In addition to the NDE Portal, every JBoss server within the DHS (ingest/factory/product generation/distribution) maintain active logs. These logs record all activity on the server and capture any warnings and/or errors from NDE applications. All server logs can be found locally in the $logdir (/opt/apps/ots/JBoss/JBoss-soa-p-5/JBoss-as/server/nde\_op1/log) directory. They are stored indefinitely with new files created daily at 0Z. The server.log is the current log file.

All NOAA Unique Products (NUP) Delivered Algorithm Packages (DAPs) are required to generate a log for each instance of execution of a production job. This log is scanned by the NDE DHS for any warnings or errors which, if found, are recorded in the database. If an algorithm fails, the working directory which includes the log file are compressed and stored in a forensics folder for later analysis. The forensics folder is located on the locally on the processing server in /opt/data/nde/NDE\_OP1/pgs/forensics\_data.

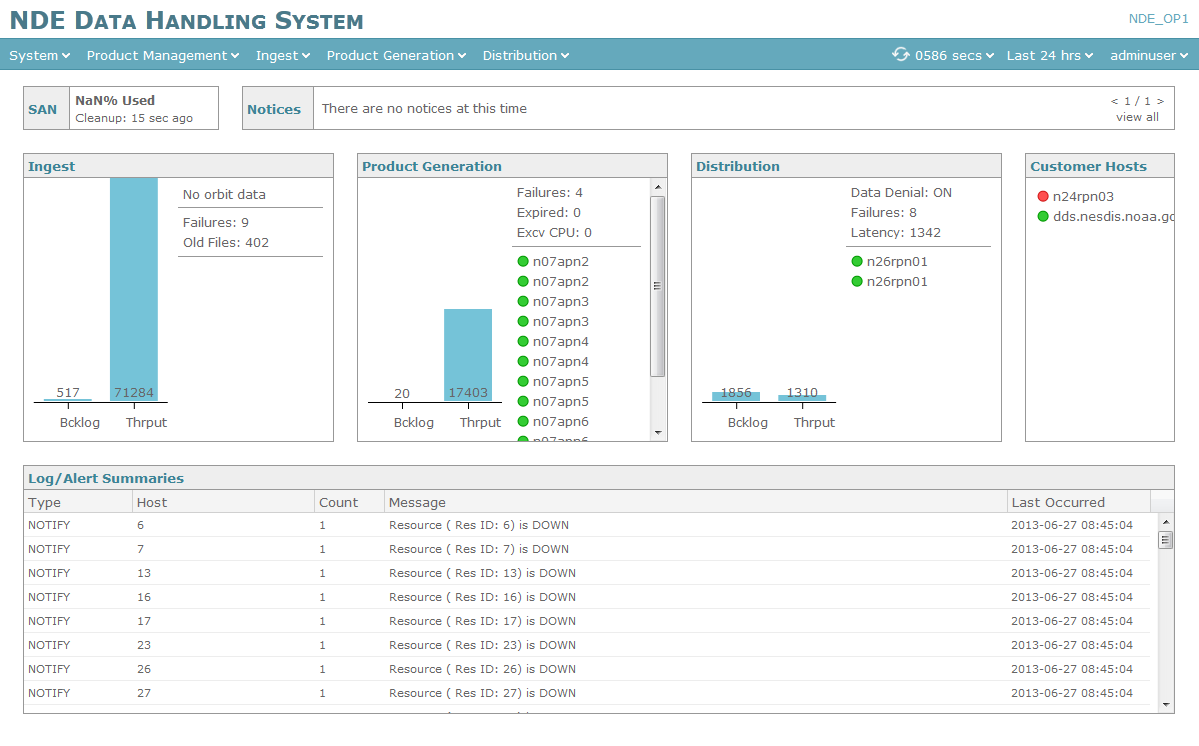


Figure ‑-1 NDE DHS Dashboard GUI Page

Ingest Monitoring

Monitoring of ingest can be accomplished using the Ingest Backlog and throughput page (below). This page shows the backlog in the landing zone (incoming\_input directory on the SAN) and any files in the ingest buffer table (post LIFO sorting by the ingest throttle but pre ingest by the ingest JBoss server). Also depicted is a list of every registered product by product shortname and the time of the last ingested file for that product. It also shows the observation time of the last ingested file along with the number of files ingested by each server.

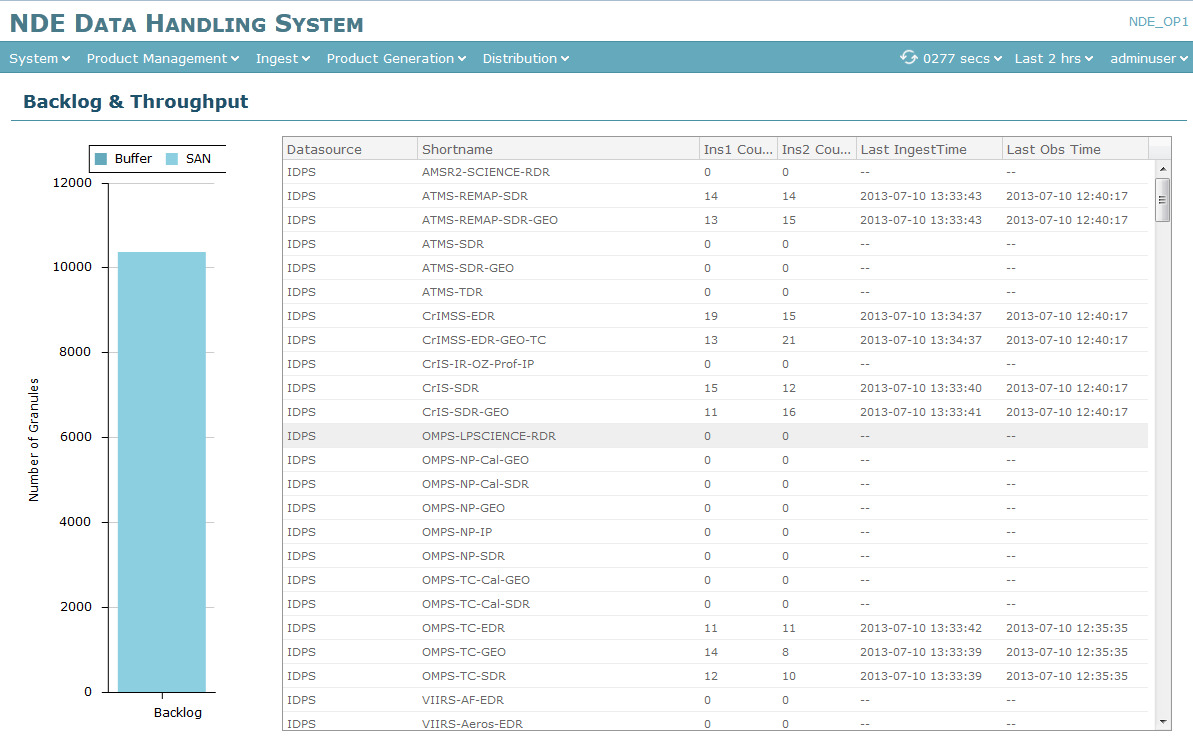


Figure ‑ NDE DHS Backlog and Throughput GUI Page

Product Generation Monitoring

Product generation is defined by a single or set of production rules for each algorithm (science and NDE Data Selection Services). The NDE DHS is data driven, therefore every production rule defines a single “trigger” product that, upon ingest, will cause the product generation factories to instantiate a production job spec for that particular production rule. The production job spec has an initial state of “INCOMPLETE” and represents a potential production job. The actual production job is not created and made available to the processors until the production job spec is put into a state of “COMPLETE”. This doesn’t happen until all required input data has been ingested into NDE and all production rule criteria have been met (e.g. observation data is over a particular region/gazetteer). If, however, not all required data that meets the production rule criteria arrives in a predefined period of time, then the production job spec will enter a permanent state of “COMPLETE-NOINPUT” and all activities associated with that spec will end.

After a production job spec has completed, a production job will be created by the product generation factory and initialized to a state of “QUEUED”. The assigned processing nodes for the particular algorithm will be looking for production jobs that are queued. After finding a queued job and if the processing node has an available job box, the production job state will change to “COPYINPUT” while the processor copies all input data to the working directory. After all data has been successfully staged in the working directory, the production job changes to a state of “RUNNING” while the algorithm executes that particular job. The production job will remain in this state until the algorithm returns with a code or errors out. If the algorithm returns with a non-zero code or simply errors out (e.g. core dump), then the production job state will change to “FAILED”. A failed state will cause the processor to compress the working directory and all of its contents into a zip file that is copied to the forensics directory for offline analysis. If the algorithm returns with a zero code, then the state will change to “COPYOUTPUT” while the processor copies the output algorithm files to the SAN for ingest. After successful completion, the state changes to “COMPLETE”. The following tables list the various states a production job spec and production job can be in.

Table 5‑1 Production Job Spec States

|  |  |  |
| --- | --- | --- |
| State | Initialization | Transition |
| INCOMPLETE | PG Factory initiates a Production Job Spec when a trigger product file is ingested (trigger is defined in the production rule). | PG Factory checks the database for all required input data for the Production Job Spec (as defined in the production rule). Can transition to COMPLETING or COMPLETE-NOINPUT. |
| COMPLETING | PG Factory has determined that all required input data that satisfies the production rule criteria has been ingested in time. | PG Factory populates all of the JOBSPEC tables and creates the Production Job. Can transition only to COMPLETE. |
| COMPLETE | JOBSPEC tables and Production Job have been created. | Production Job Spec is done – no transition from this state. Production Job is created in a state of QUEUED. |
| COMPLETE-NOINPUT | PG Factory has determined that required input data has not been ingested within a predefined period of time (as defined in the production rule). | Production Job Spec is done – no transition from this state. |

Table 5‑2 Production Job States

|  |  |  |
| --- | --- | --- |
| State | Initialization | Transition |
| QUEUED | PG Factory has completed a production job spec and created the production job. | Processing nodes that have been assigned to the algorithm and that have available job boxes select a queued job. Can transition only to COPYINPUT or FAILED. |
| ASSINGED | Processing node has picked the production job off of the QUEUE. | Transition to COPYINPUT. |
| COPYINPUT | Processor has picked up queued job and is copying all input data from the SAN to the working directory. | Processing node successfully completes copying input data from the SAN to the working directory. Can transition to RUNNING or FAILED. |
| RUNNING | All input data has been copied to the working directory and the processor has executed the algorithm driver script. | Algorithm returns with a code. Can transition to METADATA or FAILED |
| METADATA | Algorithm has returned with a zero code and written output file names to the Process Status File (PSF). NDE DHS reads the PSF file names to be ingested. | Processing node successfully reads the output file names from the PSF and validates the file name pattern with a registered product. Can transition to FAILED or COPYOUTPUT |
| COPYOUTPUT | File names have been read from the PSF file and validated.. | Processing node successfully completes copying output data from the working directory to the SAN. Can transition to COMPLETE or FAILED. |
| COMPLETE | All output data has been successfully copied from the working directory to the SAN and the working directory is removed. | Production Job is done – no transition from this state. |
| FAILED | An error has occurred somewhere along the processing chain (e.g. algorithm returns non-zero code, copy fails). | Processor compresses the working directory into a zip file and copies it to the forensics directory for offline analysis. Production Job is done – no transition from this state. However, job can be manually re-queued if desired. |

The Product Generation drop down menu has a link to the PGS Status Summary which represents a tally of all the current states a particular production rule is in. The figure below shows the status page. All states are clickable and will bring up the specific production job specs or production jobs for that status. The Production Job Spec and Production Job Status page provides more detailed information.

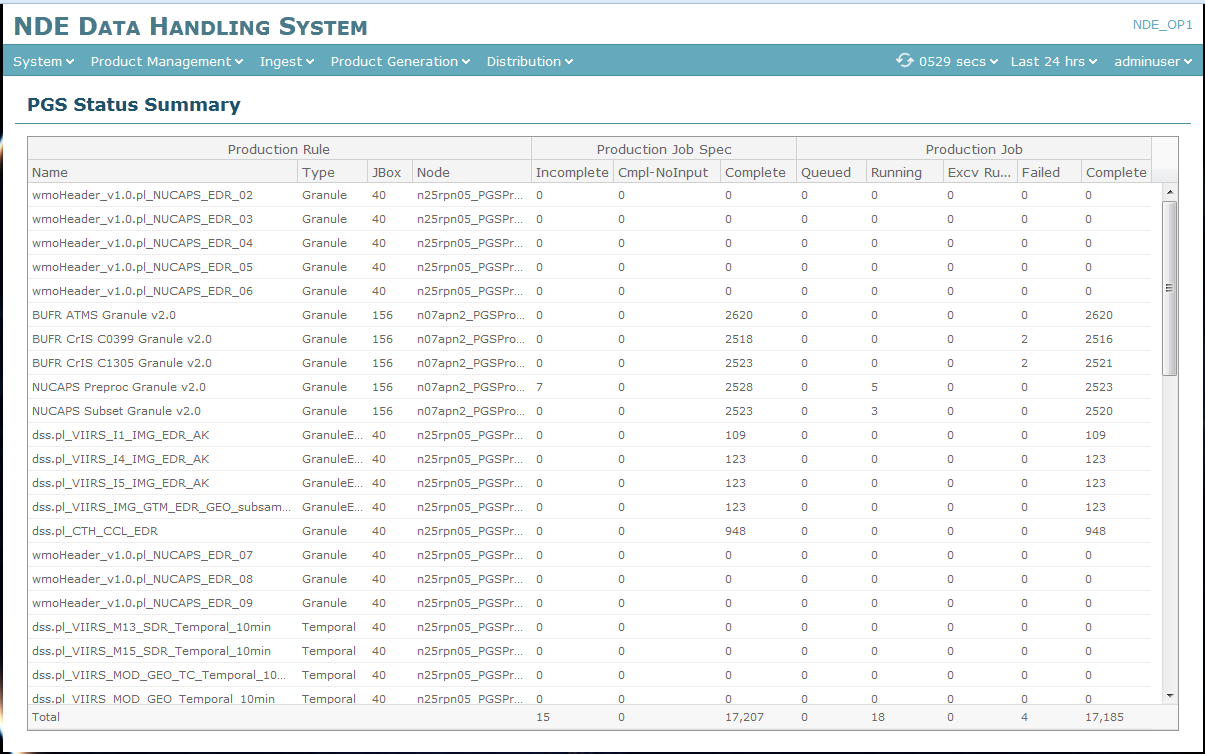


Figure ‑ NDE DHS PGS Status Summary GUI Page

The PGS Job Spec & Job Status page includes detailed information such as id numbers, observation times, start/stop times for all specs and jobs, and the statuses. Clicking on an individual job will bring up all current information on a particular job spec and/or job.

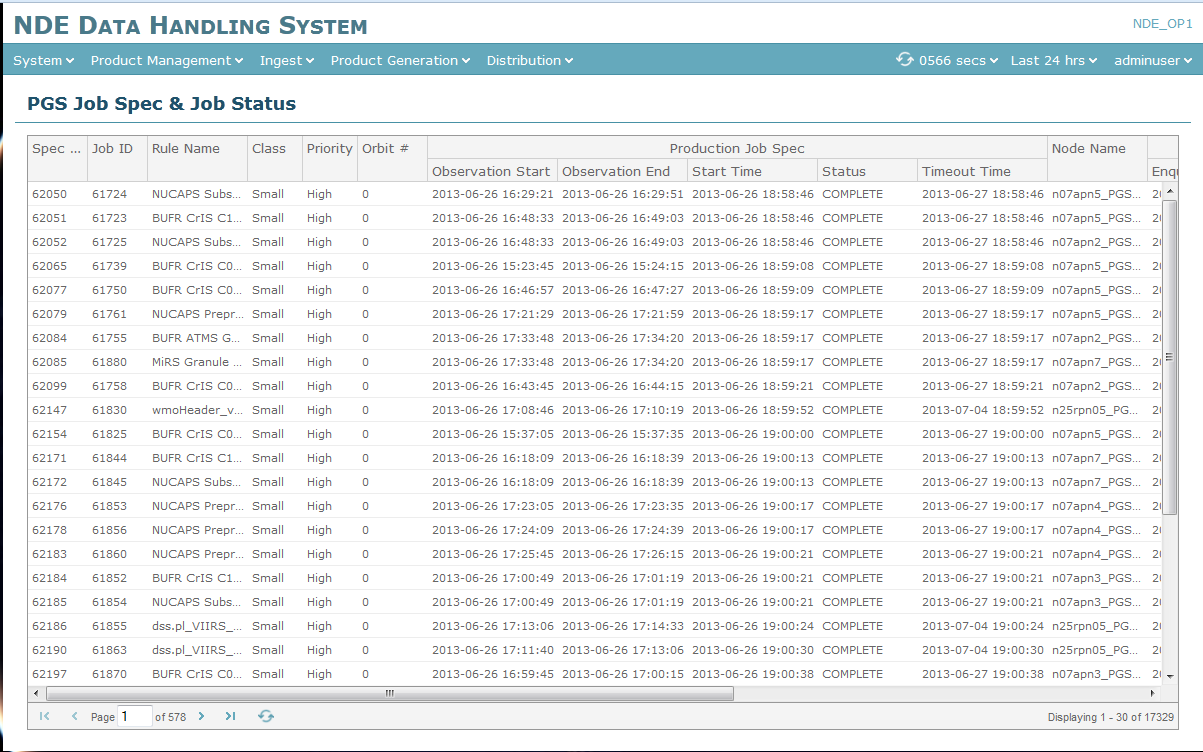


Figure ‑ NDE DHS PGS Job Spec & Job Status GUI Page

The detailed window shows all input data for a particular job (figure below). In this screenshot there are two input products listed when there should only be one, given the production rule description. This has happened because the two files have overlapping observation times. This is a known issue with this particular algorithm. This serves as an example of clicking through the GUI pages starting at the dashboard to look at a particular problem and finding the issue.

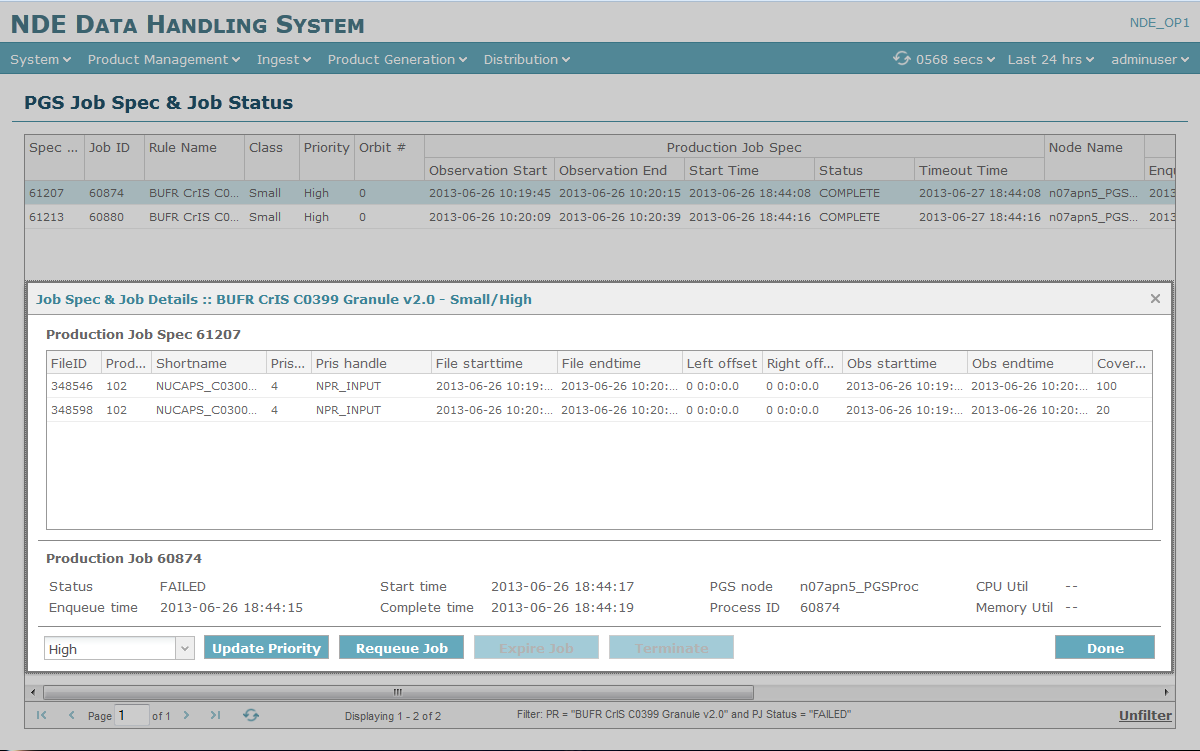


Figure ‑ NDE DHS PGS Job Spec & Job Status Details GUI Page

Product Distribution Monitoring

Product distribution begins when a user subscribed product file is ingested and the ingest server sends a message to the distribution factory. The distribution factory creates a distribution prepare request which contains all the information about a particular product file and simultaneously creates distribution requests for all users subscribed to the product. A single distribution prepare request may have one or many distribution requests associated with it. After a distribution prepare request completes, the factory creates a distribution job. The following table lists the various states for distribution prepare requests and distribution jobs.

Table 5‑3 Distribution Prepare Request States

|  |  |  |
| --- | --- | --- |
| State | Initialization | Transition |
| INITIAL | Created by the distribution factory when subscribed product is ingested and message is received from ingest server. | Distribution prepare request has been created. Transitions to PREPARING. |
| PREPARING | Factory created a distribution prepare request. | Factory sorts DPRs by priority and LIFO. Transitions to PROCESSING. |
| PROCESSING | Factory has sorted DPRs. | Factory creates links to subscribed product file in the dist directory on the SAN if it is a pull job. If defined in the subscription, the factory compresses the product file and/or creates a checksum in the dist directory. Can transition to COMPLETE or EXPIRED. |
| COMPLETE | Factory has successfully created pull links to product file in the dist directory if it is a pull job and, if applicable, compressed file and created checksum. | Distribution job is created by the factory. |
| EXPIRED | Subscription time latency threshold has been met. | No transition from this state. |

Table 5‑4 Pull Distribution Job States

|  |  |  |
| --- | --- | --- |
| State | Initialization | Transition |
| FINALIZING | Distribution prepare request is complete. | Stored procedure creates distribution job. |
| QUEUED | Distribution prepare request is complete. | Can transition to RUNNING. |
| RUNNING | Immediate transition from queued state for pull jobs. | Factory successfully creates user links to product link in dist directory. Can transition to LINKREADY or LINKFAILED. |
| LINKREADY | Factory successfully created user links. | User must FTPS pull the product to transition to PULLCOMPLETE. If pull fails or user does not pull, stays in this state. |
| LINKFAILED | Factory did not create the user link. | No transition from this state without user intervention. |
| PULLCOMPLETE | User pulls the product. | Updated by cron job script that scans the FTP log for successful pulls. |

Table 5‑5 Push Distribution Job States

|  |  |  |
| --- | --- | --- |
| State | Initialization | Transition |
| FINALIZING | Distribution prepare request is complete. | Stored procedure creates distribution job. |
| QUEUED | Distribution prepare request is complete. | Can transition to RUNNING. |
| RUNNING | Processor has picked up queued job and is attempting to push the data file. | Can transition to DELIVERED or FAILED. |
| DELIVERED | The data file has been successfully pushed to its destination. | No transition from this state. |
| FAILED | The data file has failed to be pushed to its destination. | No transition from this state without user intervention. |
| EXPIRED | Subscription time latency threshold has been met. | No transition from this state. |

### Science Maintenance

Quality monitoring is performed by the PAL’s development team and the STAR developers. Daily checks should be conducted by the PAL’s developers through their QA website on following output.

The SDR and global grids should be plotted. The plots of surface channels should be checked visually to verify that the incoming geolocation is correct. This is done by noting the spatial coherence of brightness temperature at the boundaries of land and water. Also, spatial irregularities in brightness temperatures that aren’t related to the surface geography may be manifestations of noise or other instrumental issues such as scan biases.

Time series of Principal Component Scores (PCS) and retrieval yield should be checked each day. Drops in PCS and retrieval yield will indicate abnormality in the data such as volcanic eruptions or increases in the levels of noise. These simple checks are high-level checks and are not meant to diagnose specific problems. However, if necessary, they can be used to trigger further analysis by STAR science developers.

On the STAR side the global matchup datasets will be ingested to create long-term matchup datasets between NUCAPS SDR and EDR data and radiosonde observations. These will be analyzed over different time periods to determine retrieval quality over different surface types (tropical ocean, mid-latitude land, ice, etc) and across different seasons.

STAR and OSPO personnel should communicate regularly to discuss potential data quality issues and to formulate and schedule updates to NUCAPS science code.

The following table contains the parameters which are used for generating statistics in the table format and are plotted for monitoring.

Table 5‑6 Statistics Generation

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Units | Title | Description |
| Qmean | kg/kg | Water Mean of GFS ("True") (Avg. 200 - 1100 mb) |  |
| Qbias\_trop | kg/kg | Water Bias of Full Tropo. (Avg. 200 - 1100 mb) | Bias for water vapor ratio, for accepted cases, per granule, relative to the GFS model, average for troposphere (200 to 1100 mb) |
| Qrms\_trop | kg/kg | Water RMS of Full Tropo. (Avg. 200 - 1100 mb) | RMS for water vapor ratio, for accepted cases, per granule, relative to the GFS model, average for troposphere (200 to 1100 mb) |
| Qperc\_trop | % | Water % Error of Full Tropo. (Avg. 200 - 1100 mb) | ??? |
| Tbias\_trop | K | Temperature Bias of Full Tropo. (Avg. 200 - 1100 mb) | Bias for temperature, for accepted cases, per granule, relative to the GFS model, average for troposphere (200 to 1100 mb) |
| Trms\_trop | K | Temperature RMS of Full Tropo. (Avg. 200 - 1100 mb) | RMS for temperature, for accepted cases, per granule, relative to the GFS model, average for troposphere (200 to 1100 mb) |
| Tbias\_mid | K | Temperature Bias of Mid-Tropo. (Avg. 520 - 790 mb) | Bias for temperature, for accepted cases, per granule, relative to the GFS model, average for mid-troposphere (520 to 790 mb) |
| Trms\_mid | K | Temperature RMS of Mid-Tropo. (Avg. 520 - 790 mb) | RMS for temperature, for accepted cases, per granule, relative to the GFS model, average for mid-troposphere (520 to 790 mb) |

Radiance monitoring

The key thing here is to recognize that what you want to organize the statistics so that they have the ability to monitor the detectors of the instrument. For AIRS there is 1 field-of-view (FOV) with 2378 individual detectors that maps out an arbitrary field-of-regard that was chosen to be a 3x3 configuration that mapped to an "AMSU" field-of-regard (FOR).

For IASI there is a natural 2x2 FOR as there are 4 sets of detectors. That is there are effectively 4 IASI sub-instruments that are acquired simultaneously as the scan mirror stops-and-stares at a scene. There are 3 detectors per spectrum (8461 channels) so that means the PC-scores are 12 numbers that characterize the 4 simultaneous spectra that are acquired. IASI then stops and stares 30 times in a scan line and stores these as either 22 or 23 scan sets per granule (alternating along track). Looks sort of like:

FOR.1 FOR.2 FOR.3 .... FOR.28 FOR.29 FOR.30

|-------| |-------| |-------| .... |-------| |-------| |-------|

| O O | | O O | | O O | .... | O O | | O O | | O O |

| | | | | | .... | | | | | |

| O O | | O O | | O O | .... | O O | | O O | | O O |

|-------| |-------| |-------| .... |-------| |-------| |-------|

Statistics for each granule can be computed from the 12 PC-scores per FOR can be and a bias will tell you if a detector changes and standard deviation (sdv) will tell you if that change is for all FOR's of if there is a scan dependent effect (that could happen if there is jitter in the scan motor or interferogram mirror).

For CrIS, it is the same as IASI except they have 9 FOV/FOR. Like IASI, each CrIS FOV also has 3 defectors (1305 channels). It is also more complicated because the FOR rotates as the instrument scans away from nadir. At nadir it looks like

+-------+

| O O O |

| O O O | NOTE: the outer box is same size as IASI

| O O O |

+-------+

The left most FOR is rotated -45 degrees and the right most FOR is rotated +45 deg. But the detectors within that box need to be tracked individually (that is, the upper left detector is always the same director as it is read out of the file.

A change in PC-score will tell you if the instrument is changing character (e.g, higher noise, failure of a detector, vibration of the instrument, etc.) but it also could be telling you that the Earth scene is anomalous (volcanoes, extremely cold scenes at cloud top or over polar regions, etc.).

I would expect the PC's to have orbit oscillations, day/night oscillations, or be a function of what kind of scene is underneath (desert vs. ice, etc.). It could also be a function of instrument heating - which would be a function of the solar beta angle (angle between the satellite and the Sun). So these "normal" oscillations would be a normal operating range of the PC-scores and the threshold may need to be dynamic. Initially, of course, it could be set high enough to be above the normal oscillations.

Retrieval monitoring

.out file (see example 20120509\_365\_bin\_iasi\_l2\_retrieval.out)

This file contains the following components

1. #of accepted / # to total (660 for IASI, 240 for CrIS)

#kick = # of kicked channels

has average of %land (0=ocean, 100=land), latitude, longitude,

solar zenith angle (solz), and # of scenes believe to be clear

NOTE: solar zenith is 0 overhead, 90 deg at sunrise/sunset greater than 90 indicates that sun is below the horizon this average is for entire granule so the averages looks like the center of the granule

2) formatted echo of all namelists that contain all

file names, parameters, and list of sequence steps

in the retrieval (begins with "&AIRSNL " and

ends with "&&&&&& namelist echo completed"

- one critical item is "stepname" which shows the short-name of

all the steps done in the retrieval

3) statistics on each component of the retrieval organized

by retrieval type (a "\*" indicates that GFS should contain

a reasonable value, a "x" means this is a departure from

a simple climatology and cloud be useful indicator

of interesting (i.e., variable field).

eta statistics (all zero for real data)

cloud height statistics (w.r.t. zero height clouds)

Microwave emissivity:

\* Surface skin temperature:

\* Surface pressure:

\* Surface Air Temperature

IR emissivities

IR reflectivitys

carbon dioxide statistics

OLR

COLR (clear sky OLR)

\* Temperature profiles:

- want "temp 2" column bias and rms statisics for

AV 200- 1100

AV 520- 790

\* water statistics on 1-km layers

- want "water 1" column bias, rms, and %error statistic for

AV 200- 1100

- also want 1st column from this table labeled "true"

which represents the average water (GFS) for the cases

\* water statistics on 2-km layers

- this is the same as above, but using 2-km layering

\* ozone statistics

+ methane statistics

+ carbon monoxide statistics

+ nitric acid statistics

+ nitrous oxide statisitics

+ sulfur dioxide statistics

\* geopotential height statistics

4) summary of # of clear cases by ocean, land, etc.

5) summary of convergence and avg number of iterations of each ret type

6) summary of why we rejected

7) summary of # of cases day/night land/ocean for accepted and all cases

8) timing information -- doesn't appear to be connected for ops system the accumulation of .out information

------------------------------------

for each granule we should collect the following information

the information to be extracted from

Nacc = # accepted cases, 0 .le. Nacc .le. Ntot

time = time of granule (NOT in this file)

avg{lat} = from item #1, average of latitude for all cases

avg{long} = from item #1, average of longitude for all cases

Tbias\_trop = T(p) bias of full troposphere (AV 200- 1100)

Trms\_trop = T(p) rms of full troposphere (AV 200- 1100)

Tbias\_mid = T(p) bias of mid-troposphere (AV 520- 790)

Trms\_mid = T(p) rms of mid-troposphere (AV 520- 790)

Qmean = q(p) mean of GFS ("true")

Qbias\_trop = q(p) bias of full troposphere (AV 200- 1100)

Qrms\_trop = q(p) rms of full troposphere (AV 200- 1100)

Qperc\_trop = q(p) % error of full troposphere (AV 200- 1100)

in these cases bias means

bias = (1/Nacc)\*sum{ret-gfs} for the accepted cases

rms = sqrt{(1/Nacc)\*sum(ret-gfs}^2}

See Chap.8 of airsb\_code.pdf, located at:

<ftp://ftp.orbit.nesdis.noaa.gov/pub/smcd/spb/cbarnet/reference/>

for how these are computed.

archiving the accumulation file

-------------------------------

a trigger could be set to save the rolling accumulation file

at a certain point (A.K. suggested at Eq. ascending crossing)

so that a daily file of accumulated statistics is saved

the operational system would display the accumulated file (a running

mean of last 24 hours) whereas the daily files could be concatenated

to show weekly, monthly, or annual time series of these parameters

how to generate a statistics over multiple granules

---------------------------------------------------

statistics could be generated over orbits, days, weeks, etc. or even

over other ensemble (all ascending, all descending, etc.)ascending is when solz .lt. 90 {descending is when solz .gt. 90} not exact, since granule can straddle

For IASI there are 480 granules per day, each with either 22 or 23 scan sets of 30 fields-of-regard (FOR) per scan

- they alternate so a "22 scanset" granule has 660 scenes

and a "23 scanset" granule has 690 scenes

- there are 240\*(22+23)\*30 = 324,000 scenes in a day

for NUCAPS we have "4 scanset" granules so that there are 2700

granules per day

- there are 2700\*4\*30 = 324,000 scenes in a day

for temperature

---------------

to generate a bias over a large ensemble of K granules you need to

take the bias for a single granule, bias(k), weighted by the # of

accepted cases, Nacc(k)

bias = sum{Nacc(k)\*bias(k)} / sum{Nacc(k)} where sum is for k=1,K

rms = sqrt{ ( sum{Nacc(k)\*rms(k)^2} ) / sum{Nacc(k)} }

for water

---------

water is more complicated because polar regions have significantly

less water and retrieval errors tend to be an absolute amount,

so in a percent sense polar regions have high % errors and tropics

have low % errors. So we need to weight by the amount of water.

See Chap.8 of airsb\_code.pdf, located at:

<ftp://ftp.orbit.nesdis.noaa.gov/pub/smcd/spb/cbarnet/reference/>

bias = sum{qmean(k)\*bias(k)} / sum{qmean)k)} where sum is for k=1,K

rms = sqrt{ ( sum{qmean(k)\*rms(k)^2} ) / sum{qmean(k)} }

note sure yet what to do with the %error....

... these are "living" notes and I will update as needed.

### Library Maintenance

The NDE/OSPO system administrator is responsible for keeping software libraries up to date. The system administrators follow ESPC guidelines for updating standard software libraries. Non-standard libraries such as JBoss, Perl, etc. are upgraded on a case-by-case basis. Science libraries such as Hierarchical Data Format (HDF5) must be updated on a case-by-case basis with the algorithm integrator.

### Special Procedures

Install new DHS Build – Required to implement new and repaired features of the DHS.

### Maintenance Utilities

NDE does not monitor scientific data quality. The NDE GUI provides real-time monitoring of the operational environment such as throughput/backlog for ingest, product generation, and distribution, memory and CPU usage, production job failure/success, latency, distribution. Also, the Interactive Data Language (IDL) is installed on the Linux-based processing nodes. IDL is very useful in data visualization; however, no specific tools have been developed by NDE.

## Program Backup

Currently, the IASI-CIP system provides the backup for IASI product system for the primary MetOp satellite. IASI test system provides an alternative for backing up the IASI operational system including the data required for processing.

CBU will be providing the backup for the NUCAPS production system (PE-1) including the data required for processing. Alternate back up for NUCAPS operational system can be the NUCAPS test system that is operating at the NDE. For a mission critical data one would require a hot backup in order to have a fault protected products generation. Requirements for backing up the software and data shall be on a daily basis.

# TROUBLESHOOTING

## Problem Diagnosis and Recovery

### Quality Control Output

In the NUCAPS SDR (thinned radiance) data files, all quality flags are expected to be zero when data are of “good” quality except CrIS\_QF4 below. This flag is a bit field stored in a 1-byte word. The bit in position 0 is equal is a day/night flag (0=day, 1=night time). At night the entire 1-byte word is equal to 1 if there are no other errors.

Most NUCAPS output data files contain the following six CrIS quality flags and two ATMS quality flags.

CrIS\_QF1 = QF1\_SCAN\_CRISSDR of the CrIS SDR input data.

CrIS\_QF2 = QF2\_CRISSDR of the CrIS SDR input data.

CrIS\_QF3 = QF3\_CRISSDR of the CrIS SDR input data.

CrIS\_QF4 = QF4\_CRISSDR of the CrIS SDR input data.

CrIS\_QF5 = QF1\_CRISSDRGEO of the CrIS SDR Geolocation input data.

CrIS\_QF6 = 0 reserved for future use

ATMS\_QF1 = 0, will be assigned to 1 if any relative quality flags in the CrIS SDR Geolocation input data are not equal to zero.

ATMS\_QF2 = 0, will be assigned to 1 if there is an error occurred during the ATMS resampling process.

The details of these quality flags in the input data files are presented in the [NPOESS Common Data Format Control Book](http://www.star.nesdis.noaa.gov/smcd/spb/iosspdt/qadocs/NUCAPS_CDR/D34862-03_B_CDFCB-X_Volume_III.doc) (2010). The details of ATMS resampling process are presented in the [NUCAPS Test Readiness Document (TRD)](http://www.star.nesdis.noaa.gov/smcd/spb/iosspdt/qadocs/NUCAPS_TRR/NUCAPS_TRD_Final.pptx) (NESDIS/STAR, 2010).

The NUCAPS EDR (retrieval) output data files (CCR archive and EDR) contain *Quality\_Flag* with following value settings:

0 – good

1 – rejected by physical

2 – rejected by MIT file

4 – rejected by NOAA (regression) file

8 – rejected by internal MIT

9 – rejected by physical and internal MIT

16 – rejected by internal NOAA

17 – rejected by physical and internal NOAA

24 – rejected by internal MIT and internal NOAA

25 – rejected by physical, internal MIT, and internal NOAA

### Error Correction

All NDE Data Handling System (DHS) applications are Java-based and are built using the Enterprise Service Bus (ESB) framework provided by JBoss. NDE DHS consists of five ESB applications which perform the core tasks for ingest, product generation and product distribution. Each Jboss server logs all messages, warnings, and errors in persistent log files that are refreshed every 24 hours and stored indefinitely on each Jboss server. The applications handle all exceptions with Java catch/try handlers and write appropriate messages to the log files. In addition, JBoss has built in exception handling for the ESB code. The table below lists NDE DHS application messages, possible reasons, and possible resolutions.

In the event of an algorithm error, the entire working directory for the algorithm is compressed into a forensics file and moved into the forensics directory on the local processing node for later analysis. In addition, NDE DHS scans algorithm log files for errors and warnings and logs them in the database (PRODUCTIONJOBLOGMESSAGES table).

Table 6‑1 DHS Error Messages

| Message | Subsystem Application | Reason | What to do? |
| --- | --- | --- | --- |
| executeChecks : Exception thrown | Ingest | An error occurred during the validation of a product file being ingested. Error could be during metadata extraction or when checksums don’t match. | Check to make sure h5dump (or other relevant MD extractors) is installed on the server. Make sure (manually) that the file is not corrupt. If it is determined to be corrupt, report it to the Data Provider. |
| CRC checksums did not match | Ingest | Checksum received in PAF did not match with the one generated by the App for the file being ingested | Notify Data Provider of the problem. |
| validateFilenamePattern : Unable to retrieve PAF data from message | Ingest | PAF file (\*.crc, etc.) was corrupt or empty | Same as above. |
| executeChecks : Filename is NULL | Ingest | The PAF file (.crc, etc) is incomplete and doesn’t have a File name in it | Same as above. |
| HeartbeatUpdateAction:: Error while initializing variables | Ingest | LoopName attribute may not have been defined in the Action definition (in jboss-esb.xml) | Check to make sure LoopName attribute in properly defined in the ESB configuration file (jboss-esb.xml) of the Ingest app. |
| HeartbeatUpdateAction::updateHeartBeat:: Exception caught while updating heartbeat | Ingest | Ingest app is unable to update the DB with the Heartbeat reading due to some SQL/DB error. | Check to make sure DB is functioning normally. |
| persistMetadata : Exception thrown, moving file from landing zone to failed dir | Ingest | Ingest app failed to extract metadata and/or persist it in database | Same as in row 1 above. |
| homeFile : Exception thrown routing message to queue | Ingest | JMS bridge is not working. This bridge connects Ingest and Factory apps and Ingest uses it to send messages over it to Factory app. | Check to make sure there are no errors during Ingest App deployment in JBoss server log. Check to make sure at least one Factory server is running. |
| validateFilenamePattern : Unable initialize IR/Resources objects | Ingest | Ingest app unable to retrieve rows from IngestRequestLog table for a given IR\_ID. May be a DB error or the IRB and IRL tables must be out of sync. | Make sure the DB is functioning normally. If it is, a TAL must determine the cause of the inconsistencies between IRL and IRB table. |
| validateFilenamePattern : Exception occured during validation of the product file | Ingest |  |  |
| validateProduct : Product File Unknown | Ingest | The product file being ingested is of unknown product. There DHS database doesn’t have a Product registered with the same pattern as the product file being ingested. | Make sure DHS database is properly registered with all the products. If the products registration fine, then we may have received a file that is not associated with any known registered products. |
| Rejected as duplicate | Ingest | The File being ingested has already been ingested. | Make sure no duplicate files are sent to Ingest (into incoming\_input). |
| execute :Unable to extract metadata | Ingest | H5dump or other metadata extractor has failed to extract metadata from the file. File may be corrupt or in a format that is incompatible with known format(s). |  |
| homeFile : Exception caught (moving file); | Ingest | Ingest app unable to move the file to Products ‘home’ directory after successful MD extraction and persistence. | Make sure SAN (Product Home folder) is OK. |
| validateDataFilePattern : Product Unknown | Ingest | The product file being ingested is of unknown product. There DHS database doesn’t have a Product registered with the same pattern as the product file being ingested | Make sure DHS database is properly registered with all the products. If the products registration fine, then we may have received a file that is not associated with any known registered products. |
| validateDataFilePattern : Multiple product dfs found | Ingest | Ingest app found there are multiple products registered in DB that has same filename pattern as the file being ingested. | Correct the Products registration in DB. Make sure there the filename pattern in unique in ProductsDescriotion table of DHS DB. |
| Bridge Failed to set up connections | Ingest | JMS bridge between Ingest and Factory is not correctly deployed. | Make sure the specified Factory servers in the bridge configuration are up and running. |
| createJobs: Exception detected | Dist Factory |  |  |
| Error while initializing variables | Dist Factory, PGS Factory, PGS Processor, DIST Processor | LoopName attribute may not have been defined in the Action definition (in jboss-esb.xml) | Check to make sure LoopName attribute in properly defined in the ESB configuration file (jboss-esb.xml) of the app. |
| Error while checking active flag | Dist Factory, PGS Factory, PGS Processor, DIST Processor | Apps unable to find the relevant loop ‘Active’ flags in ConfigurationRegistry table. | Make sure there are ‘Active’ flags defined in the ConfigurationRegistry table for all the loops in various apps. |
| getCompletedJobSpecs : Exception occurred | PGS Factory | App unable to execute stored proc (SP\_GET\_COMPLETED\_JOB\_SPECS) to retrieve the completed ProductionJobSpecs. May be a DB error. | Make sure the Stored Proc is properly registered in the DB and DB is functioning normally. |
| getJismoContents - Exception occurred | PGS Factory | App unable to execute stored proc (SP\_GET\_JISMO) to retrieve Job Spec Inputs. | Same as above. |
| getFileAccumulationContents - Exception occurred | PGS Factory | App unable to execute stored proc (SP\_GET\_FILE\_ACCUMULATION) to retrieve the File accumulation thresholds for a ProductionJobSpec. | Same as above. |
| createJobSpecInput : Exception detected | PGS Factory | App unable to create (insert into DB) a Job Spec Input for a ProductionJobSpec. | Make sure DB is OK. If it is, then follow the Exceptin stack trace to see which specific SQL error caused this failure. |
| updateJobStatus : Exception while updating job status | DIST,PGS Factory | App unable to update the Job status of a ProductionJob | Same as above. |
| generateChecksum : Exception caught | DIST ,PGS Factory | Apps unable to compute a checksum for a specified file. Or the app may be unable to create a checksum file. | Make sure the file in question (for which checksum is being computed) is valid and exists. Also, make sure relevant jar files (used for checksums) exists in Jboss environment. Also, make sure the folder in which the checksum file is created exists. |
| generateMD5Checksum : exception encountered | DIST Factory | Same as above for MD5 checksum. | Same as above. |
| generateCRC32Checksum : exception encountered | DIST Factory | Same as above for CRC checksum. | Same as above. |
| CompressionBean::compressFile : Exception caught | DIST Factory | App is unable to compress the file(s). | Make sure file(s) being compressed exists and are valid. If they are, follow the exception’s stack trace in the log file and determine the exact cause of the Exception. |
| createNotificationRequest : Exception caught creating NR | DIST Factory | App unable to create (insert into DB) a NotificationRequest for a given Distribution Job after it is executed. | Make sure the DB is functioning normally. If it is, follow the Exception’s stack trace to determine the root cause of the Exception. |
| NO Notification Request created for job id. The associated subscription has NO notification type. | DIST Factory | The relevant subscription (for which a job has been completed) has no Notif type defined. | Make sure a Notif type is defined for subscription in question. See appropriate documentation (User Manual) for different types of valid Notifs. |
| DistJobStatus : Exception caught | DIST Factory | The App is unable to update the status of a specified Distribution job. Usually, these updates happen as Job is moved from one state to another. | Make sure the DB is functioning normally. If it is, follow the Exception’s stack trace to determine the root cause of the problem. |
| UpdateJobStatusAction::updateNotifJobStatus : Exception occurred | DIST Factory | Same as above but for a NotificationJob. | Same as above. |
| CopyFilesAction::copyInputFiles : Exception occurred | PGS Processor | The app is unable to copy the files (may be due to problems with source and/or target folders) to PGS Job staging area. This area is first populated with the relevant input files required for executing the algorithm. | Make sure the files in question exist in Product ‘home’ directory (from which files are being copied from). Make sure the target directory can be created OK. If everything looks good, then follow the Stack trace of the exception logged to determine the root cause of the problem. |
| Exception scnning PSF file for output files | PGS Processor | The app is unable to scan the PSF file and/or create the list of output files from PSF file. PSF (Process Status File) file is generated by the Science algorithm. It lists all the Product files that have been generated by Science algorithm and are required to be ingested. | Make sure the PSF file is properly generated by Science Algorithm. If it is, then follow the Exception Stack trace to determine the root cause of error/exception. |
| JobBoxDataProviderAction::process() exception occurred | PGS Processor | This is an internal app error message logged when it is unable to manage the Job Box data in the memory. Job Box data in memory holds information on the current load (running jobs) on the processor server host. The error occurs when it is unable to update this data. | Follow the Exceptions stack trace to determine the root cause of the error. In most cases, it may require a server restart to resolve it. |
| PerformErrorHandlingAction::performErrorHandling : Exception scanning <logfilename> | PGS Processor | This error occurs when the app is unable to scan (and persists them in DB) the errors/messages from the error/log file generated by the Science Algorithm. | Make sure the Science Algorithms are generating the error/log files properly and DB is functioning normally. |
| Exception in runProdJob Action | PGS Processor | The app is unable to execute an algorithm for whatever reason. | Make sure the algorithm is properly installed on the processor. If it is, follow the exception’s stack trace to determine the root cause of the problem. |
| StartProcessingNodeAction::runLoop:: Exception retrieving jobs list | PGS Processor | This occurs when the app is unable to create a list of various Job class/size type and their number from JobBoxDataProviderSvc service. This info is used to retrieve a specific number of jobs of each class/size type. | Make sure the Job Box data is properly defined for the processor node. If it is, then an internal error might have caused this and may require a server restart to resolve it. |
| Exception occured in service RunJobSvc: UpdateJobStatus | PGS Processor | The app is unable to update the status of a Production job. The status is updated many times as it changes from one state to another. | Make sure the DB is functioning normally. If it is then follow the exception’s stack trace to determine the root cause. |
| Exception retrieving pid for algorithm command | PGS Processor | The app is unable to retrieve the PID (Process ID) of the algorithm process launched. | Determine the root cause by following the Exception’s stack trace. |
| Exception invoking stored proc LIB\_SP\_REQUESTJOB | PGS Processor | The app is unable to execute the stored proc (SP\_REQUESTJOB) to retrieve Productionjobs of a specific class/size type. | Make sure stored proc is registered in DB and DB is functioning normally. |
| notifySubscriber : FTPS Push Return not OK | DIST Processor |  |  |
| runLoop:: Exception retrieving jobs list | DIST Processor | The app is unable to execute stored proc (SP\_RETRIEVE\_JOBS) to retrieve distribution jobs from DB. | Make sure stored proc is registered in DB and DB is functioning normally. |
| Exception invoking FTPS Service | DIST Processor | This error occurs when messages are not able to be delivered to service that actually does the FTPs pushes. | Make sure the Dist processor app is deployed without any errors. If needed, redeploy the app. |
| runLoop : Exception invoking Notification Service | DIST Processor | This error occurs when messages are not able to be delivered to the service that notifies subscribers of the data availability. | Same as above. |
| updateJobBoxConfigAfterRun : Exception occurred | DIST Processor | This error occurs when the app is unable to update its internal Job Box Data after a job is executed. |  |
| Exception while updating Job Status to Fail | DIST ,PGS Processor | The app is unable to update the Job Status to Fail in DB. | Make sure database is functioning normally. If it is, then follow the stack trace in the log file to determine the root cause of exception. |
| getDistNodeId() : Exception occurred | DIST Processor | The app is unable to get the node id of the Distribution processor. Usually occurs when host on which the app is deployed and running is not configured as a distribution node. | Make sure Distribution node is properly defined in the DB. |
| transferFiles() : Connection refused | DIST Processor | This occurs when a FTPs connection is refused by the subscriber’s external host during the ftps transfers of the files. | Make sure external host data in DB is correct and that the host is able to receive ftps connections with the specified authentication. If everything is registered correctly in DHS database, then contact the concerned person on the subscriber’s side (and NDE side) and report the issue. |
| FtpsPushBean::transferFiles : Could not connect to server | DIST Processor | Same as above | Same as above |
| transferFiles : Login was unsuccessful | DIST Processor | The app is unable to login successfully into subscriber’s ftps server. | Same as above |
| transferFiles : Ftps Transfer failed | DIST Processor | FTPS transfer of files onto subscriber’s ftps server has failed. | Same as above. |
| transferFiles : Ftps Rename failed | DIST Processor | The rename (post-transfer) of files failed on subscriber’s ftps server | Same as above. |
| transferFiles : FTPConnectionClosedException | DIST Processor | The ftps connection to subscriber’s server (where files are being pushed to) has been closed unexpectedly. | Same as above. |
| transferFiles : CopyStreamException | DIST Processor | The error is logged when the data files are unable to be steamed (transferred) onto the subscribers ftps server. Usually occurs when data started to flow over ftps and then stopped for some reason before entire file is uploaded. | Same as above. |
| transferFiles : IOException | DIST Processor | This error is logged when IO problems occur during ftps transfer on the subscriber’s server where files are being pushed to. | Same as above |
| transferFiles : Unknown exception | DIST Processor | The error is reported when any unknown error caused ftps transfers to fail. | Same as above |
| notifyEmail : Exception occurred | DIST Processor | The app is unable to notify subscribers via email that the data is available. | Make sure Email (to send emails out ) is properly configured in Jboss. Make sure the email address is valid. |
| notifySoap : Exception occurred | DIST Processor | Same as above but for notifying via SOAP protocol. | Make sure the URL where SOAP message is being addressed to is correct in DB. Make sure there are no networking issues. |

### Problem Diagnosis and Recovery Procedures

Problem diagnosis and recovery procedures for the NDE DHS are listed in the Table in the preceding section. This section discusses error messages and procedures for only those errors produced by the NUCAPS system software itself.

The error output from the NUCAPS system is not coded to numerical values as in some systems. This makes it easier for developers to understand the meaning of the error in the program being developed without having to search for a numerical code in a table. Also, error messages can be more easily tailored to a given program.

Errors not described in section 6.1.3.1- 6.1.3.2 may occur in the event of hardware failure. The manifestation of hardware problems in the system operation can be very unpredictable. Operators and the technical operation staff should consider hardware problems as a possibility if large numbers of granules fail and produce messages unlike those discussed below or make reference to the UNIX operating system files. If a hardware failure is suspected, the technical operations staff should contact both OSDPD system administrators and the NUCAPS development team. The system administrators can check the machine logs to compare with the failure times presented by the operations staff. Operations should switch to the backup machine. Meanwhile, the development team can provide assistance with system recovery.

#### High-Level Errors

Table 6‑2 contains all the high-level errors that operators may encounter while checking the log files of the driver scripts for the four processing units. Each error message lists the script that produced the error as $SCRIPT (NUCAPS\_Preproc.pl, NUCAPS\_Subset.pl, NUCAPS\_Retrieval.pl, NUCAPS\_Validation.pl), the item as $ITEM (*e.g.* variable name) associated with that error, the directory as $DIR, or file/program names as $FILE, $TARGET, $PROGRAM.

The context string for these error messages is “Error in”. The NDE DHS will detect NUCAPS error messages using this context string and handle the errors accordingly.

Table 6‑2 NUCAPS Error Messages (High-Level)

| Error Message | Explanation | Action |
| --- | --- | --- |
| Error in $SCRIPT: $FILE does not exist. | The file or program does not exist in the current working directory or in the expected path. It may have a wrong path in the PCF file, have never been created, or was deleted prematurely. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT: the directory defined by $DIR does not exist. | The directory may not exist or have a wrong path in the PCF file. Or there was an unknown error in the OS. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  failed to open $FILE | The script failed to open the file or create a new file. It may have a wrong path in the PCF file, have never been created, or was deleted prematurely. Or there was an unknown error in the OS. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  failed to copy the $FILE to $TARGET. | Failed to copy the file to the target. The file may not exist or the target file name may be wrong. Or there was an unknown error in the OS. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  failed to move $FILE to $TARGET. | Failed to rename the file. The file may not exist or the target file name may be wrong. Or there was an unknown error in the OS. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  At least 1 of the 4 required ATMS files is not present. | The script will skip ATMS data files and process CrIS data files only. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  $ITEM are less than or equal to zero. | The script read a wrong $ITEM value from the PCF file. It needs to be a positive integer. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  APOD is undefined. | The script read a wrong APOD value from the PCF file. It needs to be either “yes” or “no”. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  a $ITEM string is undefined in PCF. | The $ITEM may not exist or have a wrong path in the PCF file. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  $PROGRAM returned an error. [See $FILE] | There was an error running the program or script. A log file will be appended providing additional details. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  Could NOT make link to $FILE. | Failed to make a soft-link to the file. The file may not exist or have a wrong path. Or there was an unknown error in the OS. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  Could NOT run $PROGRAM. | Failed to run a wgrib command. The command may not exist, have a wrong path, or have wrong input parameters. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  $ITEM retrieval product will not be generated | Failed to generate the EDR or CCR retrieval product. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  one or more critical input files are missing. | The required files may not exist or have wrong path names in the PCF file. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |
| Error in $SCRIPT:  Total\_Number\_of\_CrIS\_FORs in EDR GLOBAL MATCHUPS is different from SDR  GLOBAL MATCHUPS | The EDR Global Matchups failed because the total number of CrIS FORs in SDR and EDR must be equal. | If this error occurs, this error message may be forwarded to the NUCAPS development lead. |

For all the possible error messages listed in Table 6-2, there should be no need for the operator to rerun processing. Log files contain the standard output of the executable that was run inside the calling scripts. This output will contain low-level error messages and output that may only be interpretable to the NUCAPS development team.

#### Low-Level Errors

Section 6.1.3.1outlined high-level error messages returned from scripts. The origins of many of these errors, however, are from low-level sources. When this is the case, log files are added to these high-level messages containing the output of the low-level errors. These errors and their explanation are show in Table 6‑3. Each error message lists the program that produced the error as $PROGRAM (main program, subroutine, function), the item as $ITEM (*e.g.* variable name or array name) associated with that error, or a file name as file $FILE. The low-level Fortran 90 error messaging routine, *error\_messaging*, and its C++ incarnation, *error\_message*, are called thousands of times. Almost 98% of these calls are for open/close, read/write, and allocate/deallocate calls. These errors never appear to the operations staff in isolation. Most of what needs to be done in response to these errors depends on the type of response suggested by the associated high-level errors originating from the calling script.

Table 6‑3 NUCAPS Error Messages (Low-Level)

| Error Message | Explanation | Action |
| --- | --- | --- |
| FATAL ERROR: Failed to open $FILE in program $PROGRAM. | Failed to open a file. This is most often caused when a file doesn’t exist in an expected location. It may have never been created or was deleted prematurely. A parent script may assume an incorrect file name. | Depends on the action required for the high-level error table 6-1. NUCAPS developers may need to be contacted. |
| FATAL ERROR:  Failed to close $FILE in program $PROGRAM. | Failed to close a file. This is most often caused when a file doesn’t exist in an expected location. It may have been deleted prematurely or there may be an attempt to close a file that is not currently open. | Depends on the action required for the high-level error tables table 6-1. NUCAPS developers may need to be contacted. |
| FATAL ERROR:  Failed to read $FILE in program $PROGRAM. | Failed to read a file. This is most often caused when a file doesn’t exist in an expected location. It may have never been created or was deleted prematurely. | Depends on the action required for the high-level error table 6-1. NUCAPS developers may need to be contacted. |
| Error reading $ITEM from $FILE | Failed to read an item from a resource file. This is most often caused when a file doesn’t exist in an expected location. It may have never been created or was deleted prematurely. | Depends on the action required for the high-level error table 6-1. NUCAPS developers may need to be contacted. |
| FATAL ERROR:  Failed to write $ITEM in program $PROGRAM. | Failed to write to a file. This is most often caused when a file doesn’t exist in an expected location. It may have never been created or was deleted prematurely. | Depends on the action required for the high-level error table 6-1. NUCAPS developers may need to be contacted. |
| FATAL ERROR:  Failed to allocate memory for $ITEM in program $PROGRAM. | Failed to allocate memory to some data storage item. | Partly depends on the action required for the high-level error table 6-1. This most likely indicates a more serious problem with overall system resource usage. If this can be traced to a software or data format problem, the NUCAPS developers should be contacted immediately. |
| ERROR:  $PROGRAM could not allocate space for $ITEM. | Failed to allocate memory to some data storage item. | Partly depends on the action required for the high-level error table 6-1. This most likely indicates a more serious problem with overall system resource usage. If this can be traced to a software or data format problem, the NUCAPS developers should be contacted immediately. |
| FATAL ERROR:  Failed to deallocate memory for $ITEM in program $PROGRAM. | Failed to deallocate memory to some data storage item. This could occur if the memory was never allocated in the first place. | Partly depends on the action required for the high-level error table 6-1.This most likely indicates a problem with NUCAPS software design. The NUCAPS developers should be contacted immediately. |
| ERROR:  $PROGRAM could not deallocate space for $ITEM. | Failed to deallocate memory to some data storage item. This could occur if the memory was never allocated in the first place. | Partly depends on the action required for the high-level error table 6-1.This most likely indicates a problem with NUCAPS software design. The NUCAPS developers should be contacted immediately. |
| FATAL ERROR:  The number of dimensions for $ITEM is zero in program $PROGRAM. | This is an error produced by an attempt to get the number of dimensions of a variable from a NetCDF file. This error literally means what it says. The most common cause of this problem is that the NetCDF variable being sought does not exist in the file. | Depends on the action required for the high-level error table 6-1. This is a serious problem in the product design. The NUCAPS developers should be contacted immediately. |
| FATAL ERROR:  The $ITEM is a zero-length string in program $PROGRAM. | This is an error produced by an attempt to read an input file name from a resource file. This error literally means what it says. | Depends on the action required for the high-level error table 6-1. This is a serious problem in the product design. The NUCAPS developers should be contacted immediately. |
| FATAL ERROR:  The number of $ITEM (channels) and $ITEM (frequencies) is not equal in program $PROGRAM. | This means that the number of channels read out of a NetCDF file is not equal to the expected number of frequencies. This would only occur if a NetCDF template was created improperly and used in the system. | Depends on the action required for the high-level error table 6-1. This is a serious problem in the product design. The NUCAPS developers should be contacted immediately. |
| FATAL ERROR:  Failed to rewind $FILE in program $PROGRAM. | The rewind function is rarely used, but its status is always checked. This could occur if the file was prematurely close or deleted. | Depends on the action required for the high-level error table 6-1. NUCAPS developers may need to be contacted. |

The retrieval code has its own set of error definitions. These errors are linked to a code table whose contents are shown below in Table 6‑4. Any errors generated by the retrieval will revealed in the nucaps\_retrieval.log file. The granule number in which the error occurred is also presented in the message. If any of these errors occur, the STAR development team should be contacted immediately.

Table 6‑4 Retrieval Error Codes

|  |  |
| --- | --- |
| Retrieval Error Codes | Type of Error |
| 00 | Normal exit |
| 01 | Not used (the exit status is not currently used) |
| 02 | Used a bad channel |
| 03 | Option no implemented |
| 04 | Namelist error |
| 05 | I/O file error (open/close read/write) |
| 06 | Decoding input file error |
| 07 | Dimension exceeded |
| 08 | Illogical value obtained |
| 09 | Highly singular matrix |
| 10 | Tqli – too many iterations |
| 11 | Invalid physics |
| 12 | File already exists |
| 13 | Invalid symmetric inverse |
| 14 | Problems in ERR2SLAB |
| 15 | Not enough valid channels |

### Data Recovery Procedures

The NUCAPS Preproc Unit’s reader software performs range checking. Values out of physically possible ranges are set to missing. Other than that, there are no data checks to the system and no other mechanisms for removing bad data. The Subsetter Unit produces thinned radiances and passes along the original SDR quality flags without any modification. It is the user’s responsibility to check the quality flags. There is no requirement for NUCAPS to verify the quality of the thinned radiance data produced by the Subsetter Unit. The Retrieval will produce degraded retrievals if bad data enter the system. In that case, the retrieval software will reject more retrievals. This will be reflected in the retrieval quality flag. This type of event will be evident in the science quality monitoring as a decrease in the retrieval yield time series.

### Program Recovery Procedures

Algorithms are assigned across multiple nodes. If a node fails there is enough capacity so that an alternate node can continue the processing without operator intervention. If the entire system fails then job specifications can be requeued and restarted. If necessary the database will failover to the backup database.

## Application Shutdown and Restart

### Application Shutdown Procedures

The first application to shutdown is the Ingest Throttle. To do so, use the manageConfigurationRegistry.pl script in /opt/app/nde/NDE\_OP1/common and update the [host]\_IngestThrottle\_ActiveFlag to 0 (zero).

Next, shutdown the JBoss servers; This involves stopping the loops and servers themselves in a systematic fashion. For each server the loops are stopped first and then the server itself.

To stop the loops, use the manageConfigurationRegistry.pl script in /opt/app/nde/NDE\_OP1/common and update all xxxxxxLoop\_ActiveFlag parameters to 0 (zero). The loop flags are listed below:

[host]\_FTPDownloaderLoop\_ActiveFlag

[host]\_ProcessIRLoop\_ActiveFlag

[host]\_ProcessPJSLoop\_ActiveFlag

[host]\_ProcessDPRLoop\_ActiveFlag

[host]\_ProcessSubLoop\_ActiveFlag

[host]\_ProcessNRLoop\_ActiveFlag

[host]\_RetrievePJLoop\_ActiveFlag

[host]\_RetrieveDJLoop\_ActiveFlag

[host]\_RetrieveNJLoop\_ActiveFlag

To stop the JBoss servers, open PuTTY sessions to the following machines :

n25rpn[01-04] - Ingest, PGS/DIS Factory

n25rpn[05-06] - Linux-based Processor Nodes (PGS)

n07apn[2-7] - AIX-based Processor Nodes (PGS)

n26rpn[01-04] - Distribution Processor Nodes (DIS)

n26rpn[05-08] - Data Consumer Portal

n25rpn[07-08] - Data Handling System Portal (internal)

Execute the following commands on each machine, in the following machine order: Portals, Factory, Ingest, PGS Processor, DIS Processor

[host](JBoss)> cd $JBOSS\_HOME/bin

[host](JBoss)> ./manageJBossServer.pl stop nde\_op1

### Application Restart Procedures

To restart the system, first perform a shutdown as detailed in the Application Shutdown Procedures.

Starting the system involves starting the JBoss servers, the loops on the servers (if any), and the Ingest Throttle.

To start the JBoss servers, open PuTTY sessions to the following machines :

n25rpn[01-04] - Ingest, PGS/DIS Factory

n25rpn[05-06] - Linux-based Processor Nodes (PGS)

n07apn[2-7] - AIX-based Processor Nodes (PGS)

n26rpn[01-04] - Distribution Processor Nodes (DIS)

n26rpn[05-08] - Data Consumer Portal

n25rpn[07-08] - Data Handling System Portal (internal)

Execute the following commands on each machine, in the following machine order: Factory, Ingest, PGS Processor, DIS Processor, Portals

[host](JBoss)> cd $JBOSS\_HOME/bin

[host](JBoss)> ./manageJBossServer.pl start nde\_op1 [type]

Note: enter ingest, factory, processor, dmz for the [type] depending on which host the server is on. Use processor for PGS Processor Nodes and dmz for DIS Processor Nodes and Portals.

When is server starts there will be a message on the screen indicating how long it took the server to start.

JBoss (Microcontainer) [5.1.0.GA\_SOA …. ] Started in 48s:834ms

To start the loops, use the manageConfigurationRegistry.pl script in /opt/app/nde/NDE\_OP1/common and update all xxxxxxLoop\_ActiveFlag parameters to 1. The loop flags are listed below:

[host]\_FTPDownloaderLoop\_ActiveFlag

[host]\_ProcessIRLoop\_ActiveFlag

[host]\_ProcessPJSLoop\_ActiveFlag

[host]\_ProcessDPRLoop\_ActiveFlag

[host]\_ProcessSubLoop\_ActiveFlag

[host]\_ProcessNRLoop\_ActiveFlag

[host]\_RetrievePJLoop\_ActiveFlag

[host]\_RetrieveDJLoop\_ActiveFlag

[host]\_RetrieveNJLoop\_ActiveFlag

NOTE: Once a loop is started it will update the display at regular intervals.

Finally, start the Ingest Throttle. To do so, use the manageConfigurationRegistry.pl script in /opt/app/nde/NDE\_OP1/common and update the [host]\_IngestThrottle\_ActiveFlag to 1.

## System Shutdown and Restart

### System Shutdown Procedures

To shutdown a machine [rhel5 or aix]:

Login to [host] as admin user (root):

# sudo su -

# shutdown now

### System Restart Procedures

To restart a machine [rhel5 or aix]:

Login to [host] as admin user (root):

# sudo su -

# reboot

### System Reboot Procedures

To restart a machine [rhel5 or aix]:

Login to [host] as admin user (root):

# sudo su -

# reboot

# APPENDIX

## Data Flow

The various levels of the NUCAPS data flow are presented here. Section 7.1.1 presents the external interfaces to the system code. These interfaces represent the connectivity of the NUCAPS system code to the NDE DHS. Section 7.1.2 shows the next lower level of the system design. This section identifies the main system units, describes their function, and reveals their interrelationships through data flow. Section 7.1.3 describes each unit detailing its sub-unit components, their function, and sub-unit data flows.

### External Interfaces

The NUCAPS system code will be run inside of the NDE DHS. The relevant external interfaces to the NUCAPS system are shown in Figure 7‑1. This figure has been adopted from similar diagrams made available by NDE to the NUCAPS IPT. The modifications were made to mask some of the NDE DHS components that are not directly relevant to the functionality of the NUCAPS units. The requirements imposed on the NUCAPS system by NDE for the external interfaces are outlined in the NDE Product Generation Application Interface (PGAI) (NDE, 2008a).

The light blue box labeled *NUCAPS* appears as a single unit in Figure 7‑1. This box is actually a generalization of the way any one of the four NUCAPS units will function within the NDE DHS, but it appears as a single box for the sake of conceptual simplicity. The part of the NDE DHS that drives the generation of products is the Product Generation Manager (PGM).

The PGM will invoke an instance of a NUCAPS unit when all the required inputs are available on the Storage Area Network (SAN). During this invocation the DHS PGM will (1) set up a working directory, (2) change into it, (3) produce a Process Control File (PCF) in the working directory for the unit to use at run time, (4) run the unit as a system call, (5) obtain the return status when execution of the unit has completed, (6) get the PSF produced by the unit from the working directory, and then (7) collect the output from the NUCAPS run and direct it according to the status information defined in the PSF.

NUCAPS_External_Interfaces

Figure ‑ NUCAPS External Interfaces

Because the NUCAPS box in Figure 7‑1 can represent any one of the four processing units, the actual inputs for each unit will vary. For this reason, the actual types, names and formats of input and output files associated with each unit are shown later in Table 7‑2 ~ Table 7‑5 in Section 7.1.2 entitled System Level Design and Data Flow. Instead, in this section only general high-level information about inputs and outputs is provided. Below each component of Figure 7‑1 is described.

The behavior, dependencies, and resources associated with a run of each NUCAPS unit are defined by the green boxes called *Product Generation Specifications* and *System Configuration*. This is static information provided to the NDE DHS in the Delivered Algorithm Package (DAP) about the necessary inputs required to trigger a run of a given unit and what hardware resources (if any) need to be allocated.

The PCF files are supplied by the PGM to the NUCAPS unit. They will contain all the required information necessary for a particular run such as the names and locations of all the input files and ancillary data, and all the input parameters needed by the unit to generate the products. The working directory is the actual location on the disk where the run will take place. The NUCAPS unit will assume that all the necessary input files reside locally in this directory as links unless otherwise specified in the PCF.

The input data files will ultimately arrive in the NDE DHS from the Interface Data Processing Segment (IDPS) and Environmental Satellite Processing Center (ESPC). The IDPS will supply the CrIS, ATMS, and Visible Infrared Imager Radiometer Suite (VIIRS) data files. ESPC will supply the Global Forecast System (GFS) forecast files. The DHS will ingest these files and store them on the SAN. The CrIS, ATMS, and VIIRS files will arrive in HDF5 format, but will then be converted by the DHS to network Common Data Form version 4 (netCDF4) and placed on the SAN. This step is not shown explicitly in Figure 7‑11, but implicitly as arrows going from the IDPS to the SAN and then as arrows coming out of the SAN and heading to the PGM.

The execution of an individual NUCAPS unit will produce a single return code indicating status (e.g. success or failure). This will be checked by the PGM since the PGM will be performing the execution. All NUCAPS output will be placed in the working directory. The output of a NUCAPS run will consist of product files, intermediate files, and various run log files. Product files will be produced in the required format and ready for distribution such that tailoring by an external program shouldn’t be necessary. Intermediate files will be used as input for use by downstream NUCAPS units. The log files will contain information about the run status of the sub-unit components within the unit, product quality information, error messages, and standard output diagnostic messages.

Following a run of a NUCAPS unit, the PGM will make decisions about what to do with a given run’s output according to the run status of the unit and the contents of the PSF. During normal operations, the output products and intermediate files from a successful run will be directed to the SAN from the working directory by the DHS where the DHS will later redirect them for tailoring or distribution to NDE customers on a distribution server. The run logs will be sent to a *Forensics Repository* for temporary storage. These logs will be used if problems are encountered and troubleshooting is necessary.

Figure 7‑2 shows the context level data flows into and out of the NUCAPS system. The collective input and output of all the units are generalized as input and output relative to the blue NUCAPS box in the center of the figure.

NUCAPS_Context_Level_Data_Flow

Figure ‑ NUCAPS Context Level Data Flow

The arrows represent only the high-level file-based interfaces to the system, as opposed to other types of interfaces such as the unit return values, program arguments, or shell variables. Some of the file interfaces actually represent collections of files such as the *Retrieval Resource*, *CDL templates*, *Eigenvectors*, *PCF*, and *Various Run Logs*. These files are too numerous to show in a single context level diagram. Also, for any output product file listed such as the thinned netCDF4 files, there may be a few different types of subsets. More details about these collective blocks of files, and the units to which they are relevant, are provided later in section 7.1.2.

Table 7‑1 lists all NUCAPS files distributed outside of the NDE system to external users. The BUFR and AWIPS files are not produced inside the NUCAPS software, but are produced elsewhere downstream within the NDE system; the user should consult NDE and the BUFR Toolkit documents for more information on these files.

Table 7‑1 NUCAPS External Files

|  |  |  |  |
| --- | --- | --- | --- |
| File | Description | Format | Size/file |
| NUCAPS-EDR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc | This is the granule output file containing all the retrieval (profile) products. | netCDF4 | 3.1 MB/file  2700 files/day |
| NUCAPS-CCR-AR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc | This is the granule output file containing cloud-cleared radiance product data. | netCDF4 | 0.7 MB/file  2700 files/day |
| NUCAPS-OLR\_v1r0\_npp\_s???????????????\_e???????????????\_c???????????????.nc | This is the granule output file containing the outgoing longwave radiance product data. | netCDF4 | 0.6 MB/file 2700 files/day |
| NUCAPS-GG-EDR-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | CrIS/ATMS retrievals on a daily global grid at 0.5X2 degree resolution. | Gridded direct-access binary | 726 MB/file  2 files/day |
| NUCAPS-GG-OLR-GRIDS-?SC\_v1r0\_npp\_s??????????????\_e??????????????\_c??????????????.bin | Outgoing Longwave Range CrIS radiances on a daily global grid at | Gridded direct-access binary | 5.8 MB/file 2 files/day |
| NUCAPS-PCS-MONITORING\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt | This is the PCS statistics monitoring file. This is to be distributed for SDR monitoring at OSPO. | Text file | 0.0015 MB/file  2700 files/day |
| NUCAPS-EDR-MONITORING\_ v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt | This is the retrieval monitoring output file. | Text file | 0.078 MB/file  2700 files/day |
| NUCAPS\_EDR\_IUTN06\_KNES\_npp\_$.nc.wmo | NUCAPS EDR for AWIPS for 9 sectors | netCDF4 | 0.215 MB/file 1648 files/day |
| C0399\_v1r0\_npp\_s???????????????\_e???????????????\_c???????????????.bufr | The output CrIS 399-channel full spatial resolution BUFR file converted from NUCAPS netCDF4. | BUFR | 0.8 MB/file  2700 files/day |
| C1305\_v1r0\_npp\_s???????????????\_e???????????????\_c???????????????.bufr | The output CrIS 1305-channel full spatial resolution BUFR file converted from NUCAPS netCDF4. | BUFR | 2.8 MB/file  2700 files/day |

### System Level Design and Data Flow

The NUCAPS system is developed to run on the Red Hat Enterprise Linux (16 CPUs – 2GB/CPU) hardware platform. This is the hardware environment on which NDE products will be generated for NPP as specified in the NDE Critical Design Review (CDR) document (NDE, 2007). The system units and documentation will be delivered in a Delivered Algorithm Package (DAP) to the NDE Science Algorithm Development and Integration Environment (SADIE). On the SADIE, the NUCAPS units will be integrated into the NDE DHS and system tested by the NDE system development team.

The NUCAPS system level data flow is show in Figure 7‑3. The units are identified as blue boxes. There are four processing units: (1) the Preprocessor unit, (2) the Subsetter unit, (3) the Retrieval unit, and (4) the Validation unit. The inputs and outputs of each NUCAPS unit are discussed here. The white space around each unit is the domain of the NDE DHS. All output file from each unit is deposited in a local working directory by default. All input and output files from each unit are managed solely by the DHS. The data flows in the figure are somewhat simplified given the amount of space available. Therefore, greater detail about the input and output of each unit is provided in Table 7‑2~ Table 7‑5 and Figure 7‑4~ Figure 7‑8. Table 7‑2~ Table 7‑5 contain the file name, the file’s input/output designation, the files source, a description of the file (or purpose), and the *state* of the file. The file’s input/output designation is listed as *input*, *intermediate*, or *output*. To make this a little easier to visualize, the table cells are highlighted in light blue for *input*, light gray for *intermediate*, and light yellow for *output*. The file source is defined as the agent that generated the file, not the agent providing the file to the unit during execution. The agent providing all files to the NUCAPS units is the NDE DHS. The file *state* is either static or dynamic. Dynamic files are defined as those files that regularly change such as instrument data files, for example. Static files are those files that do not change. Static files may change only when there is a new delivery of the algorithm. Retrieval tuning files and CDL template files are good examples of static files.

NUCAPS_System_Level_Data_Flow

Figure ‑ NUCAPS System Level Data Flow

The details of the inputs and outputs for the Preprocessor unit, generalized in Figure 7‑3, are better described in Table 7‑2. The Preprocessing unit organizes and preprocesses all the data into a single type of intermediate granule file for the downstream units. Its main function is that of instrument data collocation. It processes temporally related sets of CrIS, ATMS, and VIIRS instrument granule data and collocates them to the CrIS Fields of Regard (FORs). The CrIS FOR is the 3X3 cluster of CrIS Fields of View (FOVs). The ATMS remapped product is already collocated and averaged to CrIS FOV number 5 within the CrIS FOR, but the ATMS FOVs and scan lines have to be matched to those of CrIS. In addition, the VIIRS cloud information (mask) stored on the VIIRS pixels are collocated and then averaged to the CrIS FOVs to derive a VIIRS cloud fraction on the CrIS FOVs. All the CrIS, ATMS, and VIIRS data from a single CrIS FOV are stored in matching arrays within a single netCDF4 file for each granule.

The incoming instrument data required by the Preprocessor unit originate from the IDPS as HDF5, but are converted inside the NDE DHS to netCDF4. The CrIS and ATMS data to be used are the Science Data Records (SDRs). The VIIRS data to be used is the Cloud Intermediate Product (IP). The unit also adds in the surface information (elevation and land fraction) for the center of each CrIS FOV from the Digital Elevation Model (DEM), and apodizes the CrIS radiances.

Table 7‑2 Preprocessor Unit Data Flow

| File | Input/  Output | Source | Description | State |
| --- | --- | --- | --- | --- |
| NUCAPS\_Preproc.pl.PCF | Input | NDE | The Process Control File supplied by the NDE PGM. | Dynamic |
| Granule\_Info | Input | NDE | A file listing the names of the CrIS, ATMS, and VIIRS files for this granule. | Dynamic |
| SCRIS\_npp\_d????????\_t???????\_e???????\_b?????\_c??????????????\_noaa\_ops.h5 | Input | NDE | CrIS granule files containing science data (radiances). | Dynamic |
| GCRSO\_npp\_d????????\_t???????\_e???????\_b?????\_c??????????????\_noaa\_ops.h5 | Input | NDE | CrIS granule files containing geolocation information for the science data. | Dynamic |
| TATMS\_npp\_d????????\_t???????\_e???????\_b?????\_c??????????????\_noaa\_ops.h5 | Input | NDE | ATMS granule files of ATMS radiances. | Dynamic |
| GATMO\_npp\_d????????\_t???????\_e???????\_b?????\_c??????????????\_noaa\_ops.h5 | Input | NDE | ATMS granule files containing geolocation information for ATMS radiances. | Dynamic |
| IICMO\_npp\_d????????\_t???????\_e???????\_b?????\_c????????????????????\_noaa\_ops.h5 | Input | NDE | VIIRS granule files containing Cloud Mask. There may be one or two of these files needed depending on the overlap with the CrIS files. | Dynamic |
| GMODO\_npp\_d????????\_t???????\_e???????\_b?????\_c????????????????????\_noaa\_ops.h5 | Input | NDE | VIIRS granule files containing the geolocation information for the Cloud Mask. There maybe be one or two of these files needed depending on the overlap with the CrIS files. | Dynamic |
| VCTHO\_npp\_d????????\_t???????\_e???????\_b?????\_c????????????????????\_noaa\_ops.h5 | Input | NDE | VIIRS granule files containing Average Cloud Top Height. There may be one or two of these files needed depending on the overlap with the CrIS files. | Dynamic |
| GCLDO\_npp\_d????????\_t???????\_e???????\_b?????\_c????????????????????\_noaa\_ops.nc | Input | NDE | VIIRS granule files containing the geolocation information for the cloud products. There may be one or two of these files needed depending on the overlap with the CrIS files. | Dynamic |
| Global\_DEM.bin | Input | NUCAPS | Digital Elevation Model file containing surface elevation and land-sea-coast coverage for the globe. | Static |
| nucaps\_all.nc | Input | NUCAPS | The netCDF4 output template. | Static |
| cris\_olr.nc | Input | NUCAPS | The netCDF4 output template for the ORL code | Static |
| resample.data | Input | NUCAPS | The CrIS weighting function coefficients which determine how averaging to be performed on the ATMS/VIIRS data collocated to CrIS | Static |
| airs\_17boxcar.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_01.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_06.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_05.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_04.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_03.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_02.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_14.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_13.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_12.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_11.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_10.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_09.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_08.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_07.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_17.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_16.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| airs\_17boxcar\_15.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_12.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_11.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_10.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_09.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_08.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_07.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_06.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_05.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_04.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_03.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_02.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_01.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_17.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_16.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_15.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_14.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| cris\_17boxcar\_13.txt | Input | NUCAPS | An ancillary data file provided by the NUCAPS delivery needed by the OLR code. | Static |
| nucaps\_preprocessor.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the preprocessor program. | Dynamic |
| cris\_geo2txt.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_cris\_geo2txt program. | Dynamic |
| viirs\_cfgeo2txt\_?.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_viirs\_cfgeo2txt program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| viirs\_cf2bin\_?.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_viirs\_cf2bin program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| viirs\_chgeo2bintxt\_1.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_viirs\_chgeo2bintxt program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| viirs\_ch2bin\_?.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_viirs\_ch2bin program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| cris\_viirs\_cf\_colocation.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_cris\_viirs\_cf\_colocation program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| cris\_viirs\_ch\_colocation.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_cris\_viirs\_ch\_colocation program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| olr.filenames | Intermediate | NUCAPS | This is the resource text file containing the input file names and input parameters for the main\_cris\_olr program. | Dynamic |
| CRIS\_GEO.txt | Intermediate | NUCAPS | The CrIS text file created by cris\_geo2txt and used by the CrIS-VIIRS collocation codes. | Dynamic |
| CRIS\_ORBIT.txt | Intermediate | NUCAPS | The CrIS text file created by cris\_geo2txt. | Dynamic |
| VIIRS\_CF\_ORBIT.txt | Intermediate | NUCAPS | The VIIRS text file created by main\_viirs\_cfgeo2txt and used by the CrIS-VIIRS Cloud Fraction collocation code. | Dynamic |
| VIIRS\_CF.bin | Intermediate | NUCAPS | The VIIRS binary file created by main\_viirs\_cf2bin and used by the CrIS-VIIRS Cloud Fraction collocation code. | Dynamic |
| VIIRS\_CH\_ORBIT.txt | Intermediate | NUCAPS | The VIIRS text file created by main\_ viirs\_chgeo2bintxt and used by the CrIS-VIIRS Cloud Height collocation code. | Dynamic |
| VIIRS\_CH\_GEO.bin | Intermediate | NUCAPS | The VIIRS binary file created by main\_ viirs\_chgeo2bintxt. | Dynamic |
| VIIRS\_CH.bin | Intermediate | NUCAPS | The VIIRS binary file created by main\_ viirs\_ch2bin and used by the CrIS-VIIRS Cloud Height collocation code. | Dynamic |
| Orbit\_Number.dump | Intermediate | NUCAPS | A temporary file used to get the Orbit number out of the CrIS HDf5 file for use in the OLR metadata. | Dynamic |
| NUCAPS\_ALL\_????????\_???????\_???????.nc | Output | NUCAPS | The output spatially and temporally collocated CrIS, ATMS, and VIIRS data. The CrIS and ATMS data file consist of radiances and the VIIRS data will consist of the cloud fraction and height averaged onto the CrIS FOVs. CrIS radiances have also been apodized. The surface information has been added in from the DEM as well. | Dynamic |
| NUCAPS-OLR\_v1r0\_npp\_s???????????????\_e???????????????\_c???????????????.nc | Output | NUCAPS | The output of the OLR code ready for distribution and archival. | Dynamic |
| Bounding\_Box\_????????\_???????\_???????.nc | Output | NUCAPS | An internal metadata file containing the bounding box and ascending/descending status for the current granule. This file is required by the Retrieval unit for generating granule-level metadata. | Dynamic |
| main\_nucaps\_preprocessor.log | Output | NUCAPS | This is the run log containing the standard output and return status of the preprocessor program. | Dynamic |
| main\_cris\_geo2txt.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_cris\_geo2txt program. | Dynamic |
| main\_viirs\_cfgeo2txt\_?.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_viirs\_cfgeo2txt program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| main\_viirs\_cf2bin\_?.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_viirs\_cf2bin program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| main\_viirs\_chgeo2bintxt\_?.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_viirs\_chgeo2bintxt program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| main\_viirs\_ch2bin\_?.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_viirs\_ch2bin program. This program maybe run 1 or 2 times based on how many VIIRS files are needed to cover the CrIS granule. The ? is replaced by a 1 or 2 based on which run used that file. | Dynamic |
| main\_cris\_viirs\_cf\_colocaiton.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_cris\_viirs\_cf\_colocation program. | Dynamic |
| main\_cris\_viirs\_ch\_colocation.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_cris\_viirs\_ch\_colocation program. | Dynamic |
| main\_cris\_olr.log | Output | NUCAPS | This is the run log containing the standard output and return status of the main\_cris\_olr program. | Dynamic |
| NUCAPS\_Preproc.pl.log | Output | NUCAPS | This is the run log containing the standard output and return status of the Preprocessor unit. | Dynamic |
| NUCAPS\_Preproc.pl.PSF | Output | NUCAPS | This is the Process Status File which is the formatted output status for the entire Preprocessor unit | Dynamic |

The Subsetter unit resides downstream of the Preprocessor unit. The Subsetter unit’s inputs and outputs are shown in Table 7‑3. The unit requires the CrIS/ATMS/VIIRS netCDF4 granule file produced by the Preprocessor unit as a dynamic data input. It reads these files, thins the CrIS coverage both spatially and spectrally, generates principal components, and writes the output into netCDF4. The netCDF4 files used for NDE defined products are converted to Binary Universal Form for the Representation of meteorological data (BUFR) product files outside of NUCAPS by the NDE DHS. The conversion is performed by the netCDF4 to BUFR product tailoring tool. The currently defined NUCAPS thinned radiance netCDF4 files is the all CrIS FOV per FOR for 399 channels. The content and format of these files is defined by the CDL templates, which are also provided as an input to the unit. The Subsetter unit also produces some additional netCDF4 output files as intermediate files which are required by the Validation unit downstream. These files also contain thinned coverage data and principal components.

Table 7‑3 Subsetter Unit Data Flow

| File | Input/  Output | Source | Description | State |
| --- | --- | --- | --- | --- |
| NUCAPS\_Subset.pl.PCF | Input | NDE | The Process Control File supplied by the NDE PGM. | Dynamic |
| NUCAPS\_ALL\_????????\_???????\_???????.nc | Input | NUCAPS (Preprocessor unit) | The full spectral and spatial resolution netCDF4 files of spatially and temporally collocated CrIS, ATMS, and VIIRS granule files. CrIS radiances have been apodized.. | Dynamic |
| eigvec\_????????\_full\_ascii\_real | Input | STAR | The full-band (for 1305 channels) eigenvector file used to generate the principal components. | Static |
| eigvec\_????????\_band1\_ascii\_real | Input | STAR | The band-1 eigenvector file used to generate the principal components. | Static |
| eigvec\_????????\_band2\_ascii\_real | Input | STAR | The band-2 eigenvector file used to generate the principal components. | Static |
| eigvec\_????????\_band3\_ascii\_real | Input | STAR | The band-3 eigenvector file used to generate the principal components. | Static |
| nucaps\_c0300\_allfovs.nc | Input | STAR | The thinned radiance netCDF4 template for all FOVs on all FORs, 399 channel. | Static |
| nucaps\_c0300\_centerfov.nc | Input | STAR | The thinned radiance netCDF4 template for the center FOV on all FORs, 399 channel. | Static |
| nucaps\_c1317\_1scan.nc | Input | STAR | The thinned radiance netCDF4 template for subsets with full resolution CrIS data on only 1 scan per granule. | Static |
| nucaps\_c0300\_centerfov\_pcs1b.nc | Input | STAR | The principal component netCDF4 template for the center FOV, full-band, 100 PCS. | Static |
| nucaps\_c0300\_centerfov\_pcs3b.nc | Input | STAR | The principal component netCDF4 template for the center FOV, 3-band, 100 PCS. | Static |
| nucaps\_c0300\_allfovs\_pcs1b.nc | Input | STAR | The principal component netCDF4 template for all FOVs, full-band, 100 PCS. | Static |
| nucaps\_c0300\_allfovs\_pcs3b.nc | Input | STAR | The principal component netCDF4 template for all FOVs, 3-band, 100 PCS. | Static |
| nucaps\_subset.filenames.c0300\_allfovs | Intermediate | NUCAPS | This is the resource file needed to generate the CrIS/ATMS/VIIRS all FOVs 399 channel file. | Dynamic |
| nucaps\_subset.filenames.c0300\_centerfov | Intermediate | NUCAPS | This is the resource file needed to generate the CrIS/ATMS/VIIRS center FOV 399 channel file. | Dynamic |
| nucaps\_subset.filenames.c1317\_1scan | Intermediate | NUCAPS | This is the resource file needed to generate the CrIS/ATMS/VIIRS 1-scan file. | Dynamic |
| nucaps\_subset.filenames.c0300\_centerfov\_pcs1b | Intermediate | NUCAPS | This is the resource file needed to generate the PCS center FOV CrIS/ATMS/VIIRS full-band file. | Dynamic |
| nucaps\_subset.filenames.c0300\_centerfov\_pcs3b | Intermediate | NUCAPS | This is the resource file needed to generate the PCS center FOV CrIS/ATMS/VIIRS 3-band file. | Dynamic |
| nucaps\_subset.filenames.c0300\_allfovs\_pcs1b | Intermediate | NUCAPS | This is the resource file needed to generate the PCS all FOVs CrIS/ATMS/VIIRS full-band file. | Dynamic |
| nucaps\_subset.filenames.c0300\_allfovs\_pcs3b | Intermediate | NUCAPS | This is the resource file needed to generate the PCS all FOVs CrIS/ATMS/VIIRS 3-band file. | Dynamic |
| pc\_scroes\_stats.filenames | Intermediate | NUCAPS | This is the resource file needed to run the PC Scores Stats code. | Dynamic |
| NUCAPS\_C0300\_ALLFOVS\_PCS3B\_????????\_???????\_???????.nc | Intermediate | NUCAPS | The CrIS PCS full spatial netCDF4 granule file for all FOVs 3-band of 100 PCS. The collocated ATMS and VIIRS data will also be included. | Dynamic |
| main\_nucaps\_subset.c0300\_allfovs.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS all FOVs 399 channel file. | Dynamic |
| main\_nucaps\_subset.c0300\_centerfov.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS center FOV 300 channel file. | Dynamic |
| main\_nucaps\_subset.c1317\_1scan.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS 1-scan run log file. | Dynamic |
| main\_nucaps\_subset.c0300\_centerfov\_pcs1b.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS center FOV 1-band file. | Dynamic |
| main\_nucaps\_subset.c0300\_centerfov\_pcs3b.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS center FOV 3-band file. | Dynamic |
| main\_nucaps\_subset.c0300\_allfovs\_pcs1b.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS all FOVs 1-band file. | Dynamic |
| main\_nucaps\_subset.c0300\_allfovs\_pcs3b.log | Output | NUCAPS | This is the run log for the CrIS/ATMS/VIIRS all FOVs 3-band file. | Dynamic |
| main\_pc\_scores\_stat.log | Output | NUCAPS | This is the run log containing the standard output and return status of the PC scores code. | Dynamic |
| NUCAPS\_Subset.pl.log | Output | NUCAPS | This is the run log containing the standard output and return status of the subsetter unit. | Dynamic |
| NUCAPS\_C0300\_ALLFOVS\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS/ATMS/VIIRS netCDF4 granule file for 399 channels on all CrIS FOVs for all FORs. This is will be converted to BUFR outside of NUCAPS | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS/ATMS/VIIRS netCDF4 granule file for 399 channels on only the center CrIS FOV for all FORs. This will be used downstream by the Validation unit. | Dynamic |
| NUCAPS\_C1317\_1SCAN\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS/ATMS/VIIRS netCDF4 granule file with only 1 full resolution scan of CrIS FOVs for all 1305 channels. This file is used in the validation unit to generate a thinned coverage daily global file. | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_PCS1B\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS PCS full spatial resolution netCDF4 granule file for the center FOV full-band of 100 PCS. The collocated ATMS and VIIRS data will also be included. This is used by the downstream Validation unit, but it is also a product itself. | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_PCS3B\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS PCS full spatial resolution netCDF4 granule file for the center FOV 3-bands of 100 PCS. The collocated ATMS and VIIRS data will also be included. This is used by the downstream Validation unit, but it is also a product itself. | Dynamic |
| NUCAPS\_C0300\_ALLFOVS\_PCS1B\_????????\_???????\_???????.nc | Output | NUCAPS | The CrIS PCS full spatial netCDF4 granule file for all FOVs full-band of 100 PCS. The collocated ATMS and VIIRS data will also be included. This is used by the downstream Validation unit. | Dynamic |
| NUCAPS-PCS-MONITORING\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt | Output | NUCAPS | This is the PCS statistics monitoring file. This is to be distributed for SDR monitoring at OSPO. | Dynamic |
| NUCAPS\_Subset.pl.PSF | Output | NUCAPS | This is the Process Status File containing the formatted status output for the entire Subsetter unit | Dynamic |

The Retrieval unit’s inputs and outputs are shown in Table 7‑4. It resides downstream of the Preprocessor unit. Although it only requires two dynamic input files, it is the most complicated unit with respect to data flows given the number of static Retrieval resource files. The two dynamic inputs are the CrIS/ATMS/VIIRS netCDF4 file from the Preprocessor unit and the GFS 6-hour forecast GRIB file. The first static inputs are the Retrieval and CCR CDL template files. These input static files define the structure of the output netCDF4 Retrieval and CCR product files. The Retrieval unit runs a retrieval algorithm executable. This executable requires a number of static retrieval resource files. There are three output files, the netCDF4 Retrieval and CCR output files, and the run log for the entire unit.

Table 7‑4 Retrieval Unit Data Flows

| File | Input/  Output | Source | Description | State |
| --- | --- | --- | --- | --- |
| NUCAPS\_Retrieval.pl.PCF | Input | NDE | The Process Control File supplied by the NDE PGM. | Dynamic |
| NUCAPS\_ALL\_????????\_???????\_???????.nc | Input | NUCAPS (Preprocessor unit) | The spatially and temporally collocated CrIS, ATMS, and VIIRS granule data files. CrIS radiances have been apodized. | Dynamic |
| gfs.t??z.pgrbf?? | Input | NCEP | The GFS 6-hour forecast file in GRIB2 format | Dynamic |
| Bounding\_Box\_????????\_???????\_???????.txt | Input | NUCAPS (Preprocessor unit) | This is an internal metadata file containing the bounding box and ascending/descending status for the current granule. This file is required for the granule-level metadata. | Dynamic |
| nucaps\_l2.nc | Input | STAR | This is the netCDF4 retrieval output template file. | Static |
| nucaps\_ccr.nc | Input | STAR | This is the netCDF4 CCR output template file. | Static |
| nucaps\_ccr\_archive.nc | Input | STAR | This is the netCDF4 CCR archive output template file. | Static |
| NUCAPS retrieval metadata template | Input | STAR | This is the granule metadata template for the NUCAPS retrieval product in netCDF4. | Static |
| NUCAPS CCR metadata template | Input | STAR | This is the granule metadata template for the NUCAPS cloud-cleared radiance product in netCDF4 | Static |
| cris\_20120515\_v10a.asc | Input | STAR | Tuning coefficient file. | Static |
| 120214\_cris\_tuning\_mask.asc | Input | STAR | Tuning mask file. | Static |
| cris\_rtaerr\_v10a.asc | Input | STAR | Radiative transmittance error file. | Static |
| airs\_olr.dat | Input | STAR | Radiative transmittance coefficients to compute Outgoing Longwave Radiation | Static |
| tr\_atms\_new.dat | Input | STAR | Most recent version of ATMS transmittance coefficients | Static |
| binary.trcoef.cris.v10a | Input | STAR | Post-flight CrlS RTA coeffieicnets | Static |
| cris\_v10a.t1 | Input | STAR | CrIS cloud averaging table | Static |
| jpl\_100.inp | Input | STAR | Ensemble error estimate of climatology | Static |
| L2.M.cov100av.v1.0.0.anc | Input | STAR | Microwave retrieval error covariance file | Static |
| L2.M.ecof\_705.v1.0.0.anc | Input | STAR | Microwave retrieval error covariance file | Static |
| L2.M.weight.hsb.v1.0.0.anc | Input | STAR | HSB weighting file | Static |
| L2.uars\_clim.v1.0.3.anc | Input | STAR | UARS climatology file for upper atmosphere | Static |
| ncep\_clim.bin | Input | STAR | NCEP climatology file for Temperature and water vapor. | Static |
| clouds\_cris.nl | Input | STAR | Cloud files name list | Static |
| io\_cris.nl | Input | STAR | Input/Output name list | Static |
| microw\_cris.nl | Input | STAR | Microwave file name list | Static |
| ozone\_cris.nl | Input | STAR | Ozone file name list | Static |
| pro\_cris.nl | Input | STAR | Profile file name list | Static |
| temp\_cris.nl | Input | STAR | Temperature file name list | Static |
| water\_cris.nl | Input | STAR | Water vapor file name list | Static |
| tobin120120.dat | Input | STAR | Noise file for CrIS channels | Static |
| atms\_1.dat | Input | STAR | Noise file for ATMS channels | Static |
| cris\_solar\_v10a.txt | Input | STAR | Solar irradiance file for the radiance calculation | Static |
| L2.masuda.v2.0.0.anc | Input | STAR | Coefficients for the Masuda surface emissivity model for ocean | Static |
| L2.I.cleartest\_coef.v2.0.2.day.anc | Input | STAR | Daytime coefficients file | Static |
| L2.I.cleartest\_coef.v2.0.2.night.anc | Input | STAR | Nighttime coefficients file | Static |
| cris\_v03.eig | Input | STAR | NOAA IR regression radiance eigenvector file | Static |
| cris\_v03.frq | Input | STAR | NOAA IR regression frequency file | Static |
| cris\_v03.reg | Input | STAR | NOAA IR regression coefficient file | Static |
| cris\_????????\_???????\_???????.binary | Intermediate | NUCAPS | The CrIS retrieval input format binary file. | Dynamic |
| atms\_????????\_???????\_???????.binary | Intermediate | NUCAPS | The ATMS retrieval input format binary. | Dynamic |
| avntest\_????????\_???????\_???????.bin | Intermediate | NUCAPS | The GFS retrieval input format binary. | Dynamic |
| ????????\_???????\_???????\_bin\_0002.ret | Intermediate | NUCAPS | The retrieval output binary. | Dynamic |
| ????????\_???????\_???????\_bin\_0002.fg | Intermediate | NUCAPS | The first guess output binary. | Dynamic |
| ????????\_???????\_???????\_bin\_0002.mit | Intermediate | NUCAPS | The MIT retrieval output binary. | Dynamic |
| ????????\_???????\_???????\_bin\_0002.ccr | Intermediate | NUCAPS | The CCR output binary. | Dynamic |
| ????????\_???????\_???????\_bin\_0002.out | Intermediate | NUCAPS | The retrieval diagnostic output file. | Dynamic |
| nucaps\_l2\_binary.filenames | Intermediate | NUCAPS | The Ret\_Prep resource file required to reformat the satellite data into the retrieval input format. | Dynamic |
| preprocessor.cfg | Intermediate | NUCAPS | The GFS\_Prep resource file required to reformat the GFS surface pressure data into the retrieval input format. | Dynamic |
| nucaps\_retrieval.filenames | Intermediate | NUCAPS | The resource file required to run the retrieval. | Dynamic |
| l2.ret.filenames | Intermediate | NUCAPS | The resource file required to reformat the retrieval output into netCDF4. | Dynamic |
| l2.fg.filenames | Intermediate | NUCAPS | The resource file required to reformat the FG output into netCDF4. | Dynamic |
| l2.mit.filenames | Intermediate | NUCAPS | The resource file required to reformat the MIT output into netCDF4. | Dynamic |
| ccr.filenames | Intermediate | NUCAPS | The resource file required to reformat the CCR output into netCDF4 format. | Dynamic |
| Metadata gen. resource | Intermediate | NUCAPS | The resource file required to generate the Retrieval and CCR metadata files by the metadata generator script. | Dynamic |
| Orbit\_Number.dump | Intermediate | NUCAPS | A text file containing the Orbit Number as dumped via h5dump from the nucaps\_all netCDF file. It is used for creating the metadata in the output netCDF files. |  |
| main\_nucaps\_netcdf\_to\_l2\_binary.log | Output | NUCAPS | The Ret\_Prep run log file. | Dynamic |
| preprocessor.log | Output | NUCAPS | The GFS\_Prep run log file. | Dynamic |
| nucaps\_retrieval.log | Output | NUCAPS | The Retrieval run log file. | Dynamic |
| main\_nucaps\_l2\_to\_netcdf.ret.log | Output | NUCAPS | The Ret\_to\_netCDF run log file. | Dynamic |
| main\_nucaps\_l2\_to\_netcdf.fg.log | Output | NUCAPS | The Fg\_to\_netCDF run log file. | Dynamic |
| main\_nucaps\_l2\_to\_netcdf.mit.log | Output | NUCAPS | The Mit\_to\_netCDF run log file. | Dynamic |
| main\_nucaps\_ccr\_to\_netcdf.log | Output | NUCAPS | The CCR\_to\_netCDF run log file. | Dynamic |
| NUCAPS-EDR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc | Output | NUCAPS | This is the netCDF4 granule output file containing all the retrieval products. | Dynamic |
| NUCAPS\_CCR\_????????\_???????\_???????.nc | Output | NUCAPS | This is the netCDF4 granule output file containing all the CCR product data. | Dynamic |
| NUCAPS-CCR-AR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc | Output | NUCAPS | This is the netCDF4 granule output file containing CCR archive product data. | Dynamic |
| NUCAPS-EDR-MONITORING\_ v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt | Output | NUCAPS | This is the retrieval monitoring output file. | Dynamic |
| NUCAPS\_Retrieval.pl.log | Output | NUCAPS | This is the run log containing the standard output and return status of retrieval sub-unit processes. | Dynamic |
| NUCAPS\_Retrieval.pl.PSF | Output | NUCAPS | This is the Process Status File containing the formatted status output for the entire Retrieval unit | Dynamic |

The Validation unit produces products for the STAR offline (non near-real time) science validation efforts. These are not meant to be production monitoring files, but are for off-line science monitoring and coefficient file generation. This unit resides downstream from all three of the other units. All Validation unit inputs and outputs are shown in Table 7‑5. The inputs to this unit are the netCDF4 granule files generated by the three upstream units. These are the full resolution, thinned radiance, principal component, retrieval, and cloud-cleared radiance files. The Validation unit output products are daily global files for three different types of products: (1) files of daily global observations, (2) globally gridded data, and matchup data (to radiosonde or other instrument locations). All three formats are Fortran direct access binary files. These output files do not require any additional tailoring and can therefore be distributed as is.

Global observation files are just daily global coverage files of satellite observations for an entire day’s worth of granules. Globally gridded and matchup data products are created by collocating the observations from CrIS FORs to predefined locations such as grid point coordinates or in-situ instrument measurement locations such as radiosonde launches. All the globally gridded data products are broken into two files per product such that all data from ascending orbits are stored in one file and all data from descending orbits are stored in another.

Note that in the current Phase 3 delivery, all validation products except the EDR and OLR grids have been deactivated in the PCF file. The code and capability to generate these products remains in the validation unit. However, in the Phase 4, this code will be removed.

Table 7‑5 Validation Unit Data Flow

| File | Input/  Output | Source | Description | State |
| --- | --- | --- | --- | --- |
| NUCAPS\_Validation.pl.PCF | Input | NDE | The Process Control File supplied by the NDE PGM. | Dynamic |
| NUCAPS\_ALL\_????????\_???????\_???????.nc | Input | NUCAPS (Preprocessor unit) | The spatially and temporally collocated CrIS, ATMS, and VIIRS granule data files. CrIS radiances have been apodized. | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_????????\_???????\_???????.nc | Input | NUCAPS (Subsetter unit) | The CrIS/ATMS/VIIRS netCDF4 files for the center FOV. | Dynamic |
| NUCAPS\_C1317\_1SCAN\_????????\_???????\_???????.nc | Input | NUCAPS (Subsetter unit) | The CrIS/ATMS/VIIRS netCDF4 files containing only 1 scan of CrIS FOVs per granule. | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_PCS3B\_????????\_???????\_???????.nc | Input | NUCAPS (Subsetter unit) | The CrIS PCS netCDF4 file for 3-bands of 100 PCS center FOV. The collocated ATMS and VIIRS data also reside in this file. | Dynamic |
| NUCAPS\_C0300\_CENTERFOV\_PCS1B\_????????\_???????\_???????.nc | Input | NUCAPS (Subsetter unit) | The CrIS PCS netCDF4 file for full-band of 100 PCS center FOV. The collocated ATMS and VIIRS data also reside in this file. | Dynamic |
| NUCAPS\_C0300\_ALLFOV\_PCS1B\_????????\_???????\_???????.nc | Input | NUCAPS (Subsetter unit) | The CrIS PCS netCDF4 file for full-band of 100 PCS all FOVs. The collocated ATMS and VIIRS data also reside in this file. | Dynamic |
| NUCAPS-OLR\_v1r0\_npp\_s???????????????\_e???????????????\_c???????????????.nc | Input | NUCAPS (Preproc unit) | The CrIS OLR netCDF4 file. | Dynamic |
| NUCAPS\_EDR\_v1r0\_npp\_???????????????\_e???????????????\_c??????????????.nc | Input | NUCAPS (Retrieval unit) | This is the netCDF4 granule file containing all the retrieval products. | Dynamic |
| NUCAPS\_CCR\_????????\_???????\_???????.nc | Input | NUCAPS (Retrieval unit) | This is the netCDF4 granule file containing all the CCR product data. | Dynamic |
| AVN????????????????? | Input | NCEP | These are the GFS analysis files generated at 00, 06, 12, and 18Z. | Dynamic |
| radiosonde\_matchup\_template | Input | NUCAPS | This is a static ASCII text file containing a list of in-situ measurement locations and observations times. These are mostly radiosonde observations, but could include aircraft or dropsondes as well. | Static |
| nucaps\_????????.binary | Intermediate | NUCAPS | The CrIS/ATMS/VIIRS global binary used as the input to the CrIS/ATMS/VIIRS matchups. | Dynamic |
| nucaps\_????????.binary | Intermediate | NUCAPS | The Retrieval global binary used as the input to the Retrieval matchups. | Dynamic |
| nucaps\_matchup\_records | Intermediate | NUCAPS | The matchup direct access record list produced by the CrIS/ATMS/VIIRS matchups and used by the Retrieval matchups. | Dynamic |
| GG\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the CrIS/ATMS/VIIRS  0.5X2 grids. | Dynamic |
| FG\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the CrIS/ATMS/VIIRS  3X3 grids | Dynamic |
| 1SCAN\_nucaps\_binary.filenames | Intermediate | NUCAPS | The resource file required to generate the CrIS/ATMS/VIIRS  1-scan binary. | Dynamic |
| nucaps\_binary.filenames | Intermediate | NUCAPS | The resource file required to generate the CrIS/ATMS/VIIRS  global binary which is needed for the matchups. | Dynamic |
| GG\_GFS\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the 0.5X2 GFS global grids. | Dynamic |
| nucaps\_matchups.filenames | Intermediate | NUCAPS | The resource file required to generate the CrIS/ATMS/VIIRS  matchups. | Dynamic |
| nucaps\_l2\_binary.filenames | Intermediate | NUCAPS | The resource file required to generate the retrieval global binary which is needed for the retrieval matchups. | Dynamic |
| nucaps\_l2\_matchups.filenames | Intermediate | NUCAPS | The resource file required to generate the retrieval matchups. | Dynamic |
| GG\_L2\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the retrieval  0.5X2 grids | Dynamic |
| GG\_OLR\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the OLR  0.5X2 grids | Dynamic |
| FG\_PCS1B\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the full-band PCS CrIS/ATMS/VIIRS  3X3 grids. | Dynamic |
| GG\_PCS1B\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the full-band PCS CrIS/ATMS/VIIRS  0.5X2 grids. | Dynamic |
| GG\_PCS3B\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the 3-band PCS CrIS/ATMS/VIIRS  0.5X2 grids. | Dynamic |
| GG\_CCR\_global\_grids.filenames | Intermediate | NUCAPS | The resource file required to generate the CCR  0.5X2 grids. | Dynamic |
| main\_nucaps\_global\_grids.log | Output | NUCAPS | The run log generated by the running of the CrIS/ATMS/VIIRS  0.5X2 grids program. | Dynamic |
| main\_nucaps\_fullglobal\_grids.log | Output | NUCAPS | The run log generated by the running of the CrIS/ATMS/VIIRS  3X3 grids program. | Dynamic |
| 2-scan run log | Output | NUCAPS | The run log generated by the running of the CrIS/ATMS/VIIRS  2-scan binary program. | Dynamic |
| main\_nucaps\_global\_binary.log | Output | NUCAPS | The run log generated by the running of the CrIS/ATMS/VIIRS  global binary program. | Dynamic |
| main\_gfs\_global\_grids.log | Output | NUCAPS | The run log generated by the running of the GFS  0.5X2 grids program. | Dynamic |
| main\_nucaps\_global\_matchups.log | Output | NUCAPS | The run log generated by the running of the CrIS/ATMS/VIIRS radiance matchups program. | Dynamic |
| main\_nucaps\_l2\_global\_matchup\_bin.log | Output | NUCAPS | The run log generated by the running of the retrieval binary program. | Dynamic |
| main\_nucaps\_l2\_global\_matchups.log | Output | NUCAPS | The run log generated by the running of the retrieval matchups program. | Dynamic |
| GG\_L2\_global\_grids.filenames | Output | NUCAPS | The run log generated by the running of the retrieval  0.5X2 grids program. | Dynamic |
| main\_nucaps\_fullglobal\_PCS1B\_grids.log | Output | NUCAPS | The run log generated by the running of the 1-band PCS CrIS/ATMS/VIIRS  3X3 grids program. | Dynamic |
| main\_nucaps\_global\_PCS1B\_grids.log | Output | NUCAPS | The run log generated by the running of the 1-band PCS CrIS/ATMS/VIIRS  0.5X2 grids program. | Dynamic |
| main\_nucaps\_global\_PCS3B\_grids.log | Output | NUCAPS | The run log generated by the running of the 3-band PCS CrIS/ATMS/VIIRS  0.5X2 grids program. | Dynamic |
| main\_nucaps\_ccr\_global\_grids.log | Output | NUCAPS | The run log generated by the running of the CCR 0.5X2 grids program. | Dynamic |
| main\_olr\_global\_grids.log | Output | NUCAPS | The run log generated by the running of the OLR 0.5X2 grids program. | Dynamic |
| NUCAPS-GG-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS/ATMS/VIIRS daily global grids at 0.5X2 degree grid resolution. | Dynamic |
| NUCAPS-FG-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS/ATMS/VIIRS daily global grids at 3X3 degree grid resolution. | Dynamic |
| NUCAPS-GG-PCS3B-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS 3-band principal component daily global grids at 0.5X2 degree resolution. | Dynamic |
| NUCAPS-GG-PCS1B-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS full-band principal component daily global grids at 0.5X2 degree resolution. | Dynamic |
| NUCAPS-FG-PCS1B-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS full-band principal component daily global grids at 3X3 degree resolution. | Dynamic |
| NUCAPS-GG-EDR-GRIDS-?SC\_v06-08-10\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | CrIS/ATMS retrievals on a daily global grid at 0.5X2 degree resolution. | Dynamic |
| NUCAPS-GG-CCR-GRIDS-?SC\_v06-08-10\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | Cloud-cleared CrIS radiances on a daily global grid at 0.5X2 degree resolution. | Dynamic |
| NUCAPS-GG-OLR-GRIDS-?SC\_v1r0\_npp\_s??????????????\_e??????????????\_c??????????????.bin | Output | NUCAPS | OLR on a daily global grid at 0.5X2 degree resolution. | Dynamic |
| NUCAPS-GG-GFS-GRIDS-?SC\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | A daily global coverage file of selected GFS forecast fields collocated to the same 0.5X2.0 degree grid as the CrIS/ATMS/VIIRS global grids. | Dynamic |
| NUCAPS-1SCAN-BINARY\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | This file is a CrIS global binary used solely for off-line eigenvector generation at STAR | Dynamic |
| NUCAPS-SDR-GLOBAL-MATCHUPS\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | This is a file with matches between CrIS/ATMS/VIIRS FORs and radiosonde (or other instrument) locations. | Dynamic |
| NUCAPS-EDR-GLOBAL-MATCHUPS\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.bin | Output | NUCAPS | This is NUCAPS retrieval output matched to radiosonde locations. | Dynamic |
| NUCAPS\_Validation.pl.log | Output | NUCAPS | This is the run log containing the standard output and return status of Validation unit sub-unit processes. | Dynamic |
| NUCAPS\_Validation.pl.PSF | Output | NUCAPS | This is the Process Status File containing the formatted status output for the entire Validation unit | Dynamic |

All the units described in this section produce intermediate data files and internal log files. They are not described in this section because the information in these files does not need to travel directly into or out of the units. These files are described in section 7.1.3.

### Unit Level Design and Data Flow

This section describes the general purpose, design, and data flows within each unit. Reference will be made to the files listed in Table 7‑2~ Table 7‑5 since they are input and output of these internal data flows. Each unit will consist of a single Perl script that wraps around a number of smaller programs such as compiled data handling and science code (written in Fortran 90 and C), Perl scripts, netCDF4 library utilities, and UNIX system calls. Each of the four units is independent with respect to program execution scope. For example, one unit is not calling another. Each of the four units is called by the DHS PGM when any of the given unit’s run requirements are met. Therefore, the units are only related to each other through the dependency of their external data flows which are illustrated in Figure 7‑3.

Figure 7‑4~ Figure 7‑10show the unit data flow. They have some common characteristics that can be described here. Each figure shows the flow through a unit. Each unit consists of a single Perl driver script. The boundaries of each script’s functional domain are indicated in by a box with the dark bolded lines. The blue boxes represent the unit sub-unit programs. Note that every blue sub-unit program box requires a resource file and produces a log file in addition to whatever product file it may generate. In the case of the preprocessor the sub-unit “collocation” refers to a number of programs and is shown in more detail in Figure 7‑5 Regular Unix system calls are not indicated since their locations are known universally throughout the system. The white boxes are files handled within the unit. Note that every unit receives a PCF and produces a PSF. All unit execution and output is produced in a local working directory chosen by the DHS

#### Preprocessor Unit

The Preprocessor unit performs several important functions.

* It collocates a given set of CrIS, ATMS, and VIIRS granules and packages it into a single netCDF4 granule file. This will be done such that each set of array elements containing data from a CrIS FOR will be ordered to match (in space and time) to the associated arrays of ATMS and VIIRS data. This will allow data from the three instruments to be used easily together.
* It adds the topography (surface elevation) and land-sea-coast flag to the CrIS FOVs. This is derived from a Digital Elevation Model (DEM).
* It applies a Hamming apodization to the three CrIS radiance bands.

The Preprocessor unit data flow is shown in Figure 7‑4 and Figure 7‑5. Execution is initiated by the NDE PGM when the necessary criteria for execution are met. The criteria consist of the arrival, into the NDE DHS, of the six dynamic input files identified in Table 7‑2. These are the SDR and IP files for CrIS, ATMS, and VIIRS. Another important condition is that the ATMS and VIIRS granule files must cover the same time window covered by the CrIS data file. More than one set of VIIRS files may be required if they do not completely cover the CrIS granule observation time interval. These files will all be made available to the algorithm by the DHS in the PCF at the time of execution. The script is designed to generate the collocated data for a single granule. Multiple instances of the script are required to process multiple granules.

NUCAPS_Preprocessor_Unit_Data_Flow

Figure ‑ Preprocessor Unit Data Flow

The Collocator sub-unit is actually made up of many programs and that part of Figure 7‑4 is shown in more detail in Figure 7‑5.

NUCAPS_Collocation_SubUnit_Data_Flow.emf

Figure 7‑5 Collocator Sub-Unit Data Flow

The PCF file will also make static files available such as the CDL templates (for the output subset files), the LAC, and the DEM files. The PCF file will contain the names to the files to be processed along with their full path (location in the file system). The complete contents of the PCF for the Preprocessor unit are identified in the NUCAPS ICD (NESDIS/STAR, 2008a).

The unit begins by passing the CrIS and VIIRS Geo files to the Collocator sub-unit. This is done within the Preprocessor unit by passing the locations of these instrument’s files to the Collocator resource file. The Collocator sub-unit is a Perl script that runs a series of Fortran programs. One of these programs matches the VIIRS FOVs to CrIS FOVs based on the mathematically determined projections of both instrument’s FOVs.

The output of the Collocator sub-unit isthe NUCAPS ALL netCDF file with the VIIRS Cloud Fraction and Cloud Height variable information added.

The preprocessor calls an apodization subroutine to remove the edge effects produced by the spectral decomposition on the CrIS bands. Next, another subroutine call obtains the topography and land-sea-coast flag from a DEM file for each CrIS FOV. The final step of the program writes out all the CrIS, ATMS, data to the NUCAPS ALL netcdf file.

The OLR sub-unit is also processed within the Preprocessor unit. This sub-unit creates the OLR netCDF output.

When Preprocessor unit completes, it returns a status to the PGM indicating success or failure. The output consists of those dynamic files identified in Table 7‑2. These output files are the Collocator,Preproc, and OLR sub-unit log files, the CrIS/ATMS/VIIRS and OLR netCDF4 data files, and the PSF file for the given run of the unit. The status for the generation of each output file is written into the PSF by the Preprocessor unit. The output files are collected by the DHS and handled according to the information in the PSF.

#### Subsetter Unit

The Subsetter unit resides downstream of the Preprocessor unit. It reads a given full resolution CrIS/ATMS/VIIRS netCDF4 file and thins its contents to produce a single set of netCDF4 output product files. Some of these files are used downstream by the Validation unit. Other files are directed by the DHS for tailoring into BUFR product files by the NDE tailoring tool suite. The Subsetter unit data flow is shown in Figure 7‑6.

NUCAPS_Subsetter_Unit_Data_Flow

Figure ‑ Subsetter Unit Data Flow

The Subsetter unit is executed by the NDE PGM. When a full resolution CrIS/ATMS/VIIRS granule file output from the Preprocessor unit is made available to the NDE DHS, the criteria for execution of the Subsetter unit has been met. The script processes only one granule. Multiple instances of the script are required to process multiple granules. The PCF for the Subsetter unit is generated in the working directory by the DHS prior to execution. The unit assumes that the PCF will be available in that local directory when it runs. The PCF contains the locations and names of the netCDF4 CrIS/ATMS/VIIRS file, the CDL template files that will be used to build the netCDF4 thinned product files, the eigenvector coefficient files required for principal component generation, and the location of the ncgen utility.

When the Subsetter Perl script runs, internally it cycles through a series of processing blocks which are defined by the “n=1,m loop” in Figure 7‑6. The blocks are, in fact, not part of a loop structure in the script, but are shown in the diagram as a loop to indicate that the same process is repeated multiple times, although with slight variations. Each block is responsible for producing a different thinned netCDF4 output file. The same CrIS/ATMS/VIIRS full resolution file is used as input for each block, but the conditions within the block determine which output product netCDF4 product to generate. Each output netCDF4 file is associated with a CDL input template file defining it data fields and thinning parameters. Each time a block processing block is run, an internal resource file is created for the particular product type. This is a small text file containing all the parameters required to create a product in a given block and is shown in Figure 5 as the white box above the blue subsetter box. The dotted line coming out of the eigenvector file box going to the subsetter resource box indicates that only some of the processing blocks will require those files (i.e. those generating principal components). The blue Subsetter box is the Fortran 90 executable subsetter main program. This program will generate a different thinned product or principal component file depending on the inputs in the subsetter resource file. Each run of the subsetter executable generates a unique log file containing all the standard output from that run. These log files may be used for diagnostic purposes. When all the processing blocks have finished executing, the Subsetter Perl script generates a PSF indicating the product generation status for each defined product from the unit. The Subsetter script then exits. The return status is obtained by the PGM. The output files are collected by the DHS and handled according to the information in the PSF.

#### Retrieval Unit

The Retrieval unit uses the CrIS and ATMS data to produce trace gas products, surface and cloud properties, and cloud-cleared CrIS radiances. The Retrieval unit Perl script is triggered by the PGM when a CrIS/ATMS/VIIRS full resolution file is available to the DHS by the Preprocessor unit. Also, a GFS six hour forecast standard product file must also be available for the run period covered by the CrIS/ATMS/VIIRS file. The script processes only one granule. Multiple instances of the script are required to process multiple granules. As with the other units, the Perl script obtains its PCF from the working directory. Several static files are also made available to the unit such as the LAC file, the Retrieval and CCR metadata templates, the Retrieval and CCR CDL product file templates, and the many retrieval static files highlighted in light blue in Table 7‑4. The Retrieval unit data flow is shown in Figure 7‑7.

NUCAPS_Retrieval_Unit_Data_Flow

Figure ‑ Retrieval Unit Data Flow

The unit begins by reading a full resolution CrIS/ATMS/VIIRS netCDF4 file and reformatting it into two separate CrIS and ATMS sequential access binary files that are in the required retrieval input format. This step is performed by the program identified as Ret\_Prep (retrieval preparation). .

The matching 6-hour GFS forecast file is read by the GFS\_Prep (GFS preparation) program. It extracts the GFS surface pressure at the locations of the CrIS center FOVs. The surface pressure is also output into the sequential access binary form required by the retrieval. These CrIS center FOV locations are read from the CrIS binary produced by Ret\_Prep. The CrIS/ATMS/VIIRS binary and GFS surface pressure binary are directed into the Retrieval Resource file along with the Retrieval Static files. The Retrieval program is then run. It produces a log file as well as a Retrieval output binary, a first guess binary, an MIT binary, and a CCR binary. The Retrieval, first guess, and MIT binaries are directed to the Ret\_to\_netCDF program through the Ret\_to\_netCDF resource file. The Ret\_to\_netCDF program converts the contents of its three input files into a single netCDF4 file. The CCR binary is directed to the CCR\_to\_netCDF program by the CCR\_to\_netCDF resource file. This program converts the CCR binary to a CCR netCDF4 file. Both the Retrieval and CCR netCDF4 files are output in the local working directory. The Ret\_to\_netCDF and CCR\_to\_netCDF programs each produce a log file. Metadata for the EDR and CCR files is generated by both the driver script (NUCAPS\_Retrieval.pl) and the Fortran programs (main\_nucaps\_l2\_to\_netcdf and main\_nucaps\_ccr\_to\_netcdf). It is inserted into the header of the netCDF files and follows ISO and CF version 1.5 standards. The script finishes producing a PSF file in the local working directory.

#### Validation Unit

The Validation unit data flow is shown in Figure 7‑8~ Figure 7‑10. The entire unit is run by a single Perl script indicated by the surrounding black box. However, due to the amount of content in the flow diagram, this box has been spread over the three figures. All the blue boxes are single Fortran main programs designed to produce a single type of product. Within the unit these Fortran programs run in series and not simultaneously even though most of them are not dependent on one another. These programs are consumptive of memory and CPU cycles and it is therefore better to run them in series. Furthermore, these are not products for near real-time use so they do not have the same timeliness constraints.

There are three basic types of validation products: grids, binaries, and matchups. These are daily products that are global in scope and coverage. The grids and matchups are products in which satellite observations are collocated to a set of pre-defined grid points (in the case of the gridded products), or to in-situ measurement locations and times (in the case of the matchup products). The binaries are just daily global compilations of the satellite data extracted from the granules. These can be thinned or full spatial resolution. The binary products are generated for intermediate processing or off-line use. For each binary or matchup product there is only one output data file. For each grid product there are actually two output files, one for ascending and the other for descending (orbits) observations. To simplify the flow diagrams, however, the grid outputs are represented by single boxes. The grids are generated at either 3.0X3.0 or 0.5X2.0 degrees. For the 3.0X3.0 degree grids, at each grid point, all the data from a single field of regard (all nine FOVs) is matched and saved. The match location is based on the center FOV in the field of regard (CrIS FOV #5). For the 0.5X2.0 grids, because these are higher resolution, only the center CrIS FOV is matched and saved at a grid point. In each case the near center FOV is picked for the match. All validation products are direct access binary files. The details of these file formats are outlined in the NUCAPS ICD (NESDIS/STAR, 2008a).

The three upstream units are designed to process a single granule of data. The Validation unit, however, processes an entire day’s granules at once. Therefore, the unit only runs once a day. The unit script is triggered as a function of time, as opposed to the arrival or availability of data in the DHS. It runs on the previous day’s data and runs regardless of whether all the input data are available or not. Granules are often missing from a given day’s data so it does not make sense to wait for all data to be available. Furthermore, part of the purpose of the unit is to assess the timeliness of data being made available to the NUCAPS units.

The Validation unit begins by reading the PCF file it receives at the time of execution by the PGM. As with the other units, the PCF contains the location of all input files. The first products are the CrIS/ATMS/VIIRS center FOV 0.5X2.0 degree grids. The input CrIS/ATMS/VIIRS center FOV subset netCDF4 files are produced upstream by the Subsetter unit. They are read and collocated to a 0.5X2.0 degree global grid. This task is performed by the blue box marked *CrIS/ATMS/VIIRS 0.5X2.0 grids*. These files are validation products themselves, but they are also inputs into the GFS 0.5X2.0 global forecast grids. This program is marked by the blue box labeled *GFS 0.5X2.0 grids*. It reads in the GFS forecast files and collocates the forecast data to the same 0.5X2.0 grid generated by the *CrIS/ATMS/VIIRS 0.5X2.0 grids*.

The program generates the grids as well as a log file in the working directory. The next program is the *CrIS/ATMS/VIIRS 2-scan binary* program. It creates reads a day’s worth of the 2-scan subset netCDF4 files produced by the Subsetter unit to generate a single daily file of data. Each record in the file contains all the collocated satellite data for a single CrIS FOR. The program also produces a log file. The data file and log files associated with the run are deposited in the working directory.

The next product is the CrIS/ATMS/VIIRS matchup file. A full day of the CrIS/ATMS/VIIRS full resolution netCDF4 files, produced by the Preprocess unit, are read and written out to two direct access binary files by the program represented by the blue box labeled *CrIS/ATMS/VIIRS binary* In Figure 7‑8. One binary contains all the satellite data for an entire day. The file contains only the locations and times of the satellite observations, but both files have exactly the same number of records and those records correspond to the same measurements. Both binary files are directed to the matchup program box labeled as *CrIS/ATMS/VIIRS matchups*. This matchup program also reads in a matchup location list text file. This file contains the list of all the in-situ measurement times and locations. These observations are mostly from radiosonde launches. The program then reads the location binary input file and collocates all the satellite data locations on the nearest CrIS FOR to each matchup location and time.

Each time a matchup is made the direct access record number is saved. When the matching process is finished the other binary file containing the actual satellite data is read, but only those direct access record numbers for which there were matches are actually extracted. This approach speeds up the collocation by reducing the need to read a large file many time during the matching. The output consists of a single direct access file containing all the satellite data matchups, a log file, and a file containing a list of the direct access record numbers where there were matches in the original. Note that these record numbers are used in Figure 7‑9. The CrIS/ATMS/VIIRS 3.0X3.0 degree grids also use the same input files as the matchups. As with the 0.5X2.0 grids, a day’s worth of satellite data are collocated to a predefined 3.0X3.0 grid. The output consists of an ascending and descending grids and a log file.

NUCAPS_Validation_Unit_Data_Flow_P1

Figure ‑ Validation Unit Data Flow - Part 1

The Retrieval matchups shown in Figure 7‑9require a daily binary input file created from a day’s worth of the retrieval netCDF4 files. These netCDF4 files are produced by the Retrieval unit. An important feature of this daily retrieval binary file is that it has exactly the same number of direct access records as that of the CrIS/ATMS/VIIRS binary. Furthermore, those records are ordered in the same sequence. When the retrieval matchups run, the direct access record list file generated by the CrIS/ATMS/VIIRS matchups is read in such that only those direct access records are read the from input daily retrieval binary. This means that (1) an actual collocation with the retrieval data does not have to be done, which saves time and computing resources, and (2) the retrieval and CrIS/ATMS/VIIRS matchups can be used together in validation because their records are in exact sequential correspondence. The output of the retrieval matchups consists of the direct access binary file and a run log.

NUCAPS_Validation_Unit_Data_Flow_P2

Figure ‑ Validation Unit Data Flow - Part 2

The next set of processing is for the CrIS/ATMS retrieval 3.0X3.0 and 0.5X2.0 degree grids. These products are generated by two separate programs, but they use the same set of retrieval netCDF4 files. These input netCDF4 files are generated by the Retrieval unit. The final grid set in Figure 7-9 are the OLR 0.5X2.0 degrees. The input for the grids are the CrIS OLR netCDF granule files. The output for each program consists of the gridded files and the associated run logs.

The gridded principal component files are four separate gridded product sets. There are principal components generated for 1 (all channels) and 3 bands (derived from each band of the CrIS measurement spectrum) and these are gridded at both 3.0X3.0 and 0.5X2.0 degree resolutions. The input files for these products are derived from the Subsetter unit. Both sets of input files are full spatial resolution netCDF4 granule files. Note that only two programs actually produce these products since the same gridding program can generate both the 1 and 3 band products. However, as with all processing in NUCAPS, each instance of a run generates a single run log file.

NUCAPS_Validation_Unit_Data_Flow_P3

Figure ‑ Validation Unit Data Flow - Part 3

The last block of processing is the CCR 0.5X2.0 gridded product file. The inputs for this product are the netCDF4 CCR files produced by the Retrieval unit. As with the other processing block, a set of product files and a run log are generated.

As with the other units, the actual system execution status is returned directly to the PGM after completion. All output is deposited in the local working directory. A single PSF is generated by the unit script for the run to indicate run status of the unit and its individual products.

## Input Data Files

Eight or twelve input satellite data files for each granule are required to run NUCAPS system. They are ATMS TDR Product Data file (170KB), ATMS SDR Geolocation Product Data file (160KB), CrIS SDR Product Data file (17MB), CrIS SDR Geolocation Product Data file (110KB), and one or two sets of the VIIRS CF SDR (1.7MB), VIIRS CF Geolocation (25MB), VIIRS CH SDR(352KB), and VIIRS CH Geolocation (592KB) Product Data files. . All these files are in HDF5 format.

The details of these input data files are presented in the [NPOESS Common Data Format Control Book](http://www.star.nesdis.noaa.gov/smcd/spb/iosspdt/qadocs/NUCAPS_CDR/D34862-03_B_CDFCB-X_Volume_III.doc) (2009).

The input data files are handled by NDE DHS with the following name structures.

TATMS\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

GATMO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

SCRIS\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

GCRSO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

IICMO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

GMODO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

VCTHO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

GCLDO\_npp\_d${Year}${Month}${Day}\_t${Granule\_Start}\_e${Granule\_End}\_b${Dump}\_c${ Data\_Creation }\_noaa\_ops.h5

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time HHMMSSS (hour, minute, second, and tenthof a second)

*${Granule\_End}* = the 7-digit granule end time HHMMSSS (hour, minute, second, and tenthof a second)

*${Dump}* = the 5-digit dump number

*${Data\_Creation}* = the 14-digit product file creation time YYYYMMDDHHMMSS (4-digit year, 2-digit month, day, hour, minute, second)

The full input data files list is shown in Table 7‑2~ Table 7‑5.

## Ancillary Data Files

### Digital Elevation Model

There is one Digital Elevation Model (DEM) file: *Global\_DEM.bin*

It contains the following fields: latitude, longitude, topography (elevation in meters), land fraction, and land/sea mask. The values in the file apply to the center of a grid cell. The DEM is a global file with a resolution of 21600 latitude points X 43200 longitude points. This provides a grid resolution of 0.0083° X 0.0083°. This file is static and is delivered as part of the system which is why the DEM resides in the system file directory. The file is used in the L1C Subsetter and L1B Processing units. In these units, the preprocessing for level2 adds the DEM information. The downstream Level 2 Processing unit code requires this surface information for the retrieval.

### Retrieval System Files

There are a number of static retrieval system files. These are inputs to the NUCAPS retrieval, but unlike data files, they are static and are only updated with a delivery of the system.

Table 7‑6 contains the file name in the first column and the second column contains a brief description of the file.

Table 7‑6 Retrieval System Files

|  |  |
| --- | --- |
| RTA files | |
| airs\_olr.dat | Rapid transmittance coefficients to compute Outgoing Longwave Radiation |
| tr\_atms\_new.dat | Most recent version of ATMS transmittance coefficients |
| binary.trcoef.cris.v10a | post-flight CrIS RTA coefficients |
| cris\_rtaerr\_v10a.asc | RTA error file |
| Cloud Averaging Table | |
| cris\_v10a.t1 | CrIS cloud averaging table |
| Ensemble Error Estimate File | |
| jpl\_100.inp | Ensemble error estimate of climatology |
| MIT Retrieval Files | |
| L2.M.cov100av.v1.0.0.anc | MW retrieval error covariance file |
| L2.M.ecof\_705.v1.0.0.anc | MW retrieval error covariance file |
| L2.M.weight.hsb.v1.0.0.anc | HSB weighting file |
| L2.uars\_clim.v1.0.3.anc | L2 UARS climatology file for upper atmosphere |
| ncep\_clim.bin | Binary NCEP climatology file for T(p) and q(p) |
| Name List Files for the Main Retrieval Program | |
| clouds\_cris.nl | Cloud files name list |
| io\_cris.nl | Input/Output name list |
| microw\_cris.nl | Microwave file name list |
| ozone\_cris.nl | Ozone file name list |
| pro\_cris.nl | Profile file name list |
| temp\_cris.nl | Temperature file name list |
| water\_cris.nl | Water vapor file name list |
| Noise Files | |
| tobin120120.dat | Noise file for CrIS |
| atms\_1.dat | Noise files for ATMS |
| Solar Irradiance Files | |
| cris\_solar\_v10a.txt | Solar irradiance file for the radiance calculation |
| Coefficient Files to Compute Surface Emissivity | |
| L2.masuda.v2.0.0.anc | Coefficients for the Masuda surface emissivity model for ocean |
| Clear Flag Coefficient Files | |
| L2.I.cleartest\_coef.v2.0.2.day.anc | Day time coefficients |
| L2.I.cleartest\_coef.v2.0.2.night.anc | Night time coefficients |
| Regression Coefficient Files | |
| cris\_eigvec\_05122015 | NOAA IR regression radiance eigenvector file |
| cris\_v03.frq | NOAA IR regression frequency file |
| cris\_ccr\_05142015.asc | NOAA IR regression coefficient file |
| cris\_cld\_05122015.asc | NOAA IR angle coefficient file |
| Tuning Coefficient Files | |
| cris\_20120515\_v10a\_fin.asc | Tuning coefficient file |
| 120214\_cris\_tuning\_mask.asc | Tuning mask file |

### GFS Forecast Files

These are forecast files generated by NCEP and pushed (by NCEP) to the ESPC/DDS. These files are needed by both the EDR Processing and the Global Grids unit. The transfer of these files is shown in both Figure 7‑1and Figure 7‑2.

The files have the following name structure:

gfs.t${Hour}z.pgrbf${Forecast}

where:

*${Hour}* = the time for which the forecast is run (00Z, 06Z, 12Z, and 18Z)

*${Forecast}* = the forecast projection time (in hours = 00, 03, 06, 09, and 12)

00, 03, 06, 09, and 12 hour forecasts are run every six hours. The files are GRIB2 format files and are read with the *wgrib2* reader which is freely available from NCEP. The header content of any GRIB2 file can viewed by running *wgrib2* and supplying the file name as an argument to the command.

The forecast file preprocessor in the EDR Processing unit uses these files to extract only the surface pressure. The retrieval uses the surface pressure to anchor its solution to the surface. These files are also used to generate the GFS forecast global grids by the script *Run\_NUCAPS\_Validation.pl*. This script is part of the Global Grids software unit. The following forecast variables are extracted from 91 levels and used by this processing:

Run Hour

Forecast Hour

Forecast Latitude

Forecast Longitude

Pressure

Temperature

Water Vapor

Ozone

2 meter Dew Point

2 meter Temperature

Skin Temperature

Surface Pressure

Precipitable Water Content

Total Column Ozone

Sea Surface Temperature

Land Fraction

Temperature of the 30 mb to 0 mb layer

### Eigenvector Files

There are four of these files, one for each of the three bands, and one for the entire band. These are binary big-endian files. They are not external inputs to the system, but are in fact part of the system. As such, they come with the system delivery and are automatically installed with the system by the system installation script. Any update to these files will be part of an update to the system in general.

They all have the same name structure as described below:

eigvec\_${Year}${Month}${Day}\_${Band}\_ascii\_real

where:

*${Year}* = 2-digit year

*${Month}* = 2-digit month

*${Day}* = 2-digit day

*${Band}* = the band option (full, band1, band2, and band3)

The date string indicates when the file was generated. This file contains the eigenvector coefficients required for principal component radiance reconstructions. It is a file that will need to be updated about once every six months or if there are major changes to the calibration of the instrument.

### OLR Boxcar files

The OLR code uses a number of boxcar static files that are provided with the system. These files are called:

airs\_17boxcar\_01.txt

airs\_17boxcar\_02.txt

airs\_17boxcar\_03.txt

airs\_17boxcar\_04.txt

airs\_17boxcar\_05.txt

airs\_17boxcar\_06.txt

airs\_17boxcar\_07.txt

airs\_17boxcar\_08.txt

airs\_17boxcar\_09.txt

airs\_17boxcar\_10.txt

airs\_17boxcar\_11.txt

airs\_17boxcar\_12.txt

airs\_17boxcar\_13.txt

airs\_17boxcar\_14.txt

airs\_17boxcar\_15.txt

airs\_17boxcar\_16.txt

airs\_17boxcar\_17.txt

airs\_17boxcar.txt

cris\_17boxcar\_01.txt

cris\_17boxcar\_02.txt

cris\_17boxcar\_03.txt

cris\_17boxcar\_04.txt

cris\_17boxcar\_05.txt

cris\_17boxcar\_06.txt

cris\_17boxcar\_07.txt

cris\_17boxcar\_08.txt

cris\_17boxcar\_09.txt

cris\_17boxcar\_10.txt

cris\_17boxcar\_11.txt

cris\_17boxcar\_12.txt

cris\_17boxcar\_13.txt

cris\_17boxcar\_14.txt

cris\_17boxcar\_15.txt

cris\_17boxcar\_16.txt

cris\_17boxcar\_17.txt

cris\_17boxcar.txt

olr\_reg\_coef\_cv005\_17boxcar\_2.asc

rad\_corr\_reg\_coef\_17boxcar\_airsv10ab\_2.asc

### VIIRS collocation LUT Files

The CrIS-VIIRS collocation code uses a set of look up tables to more quickly collocate the two instruments. These files are called:

CrIS\_VIIRS\_MOD.dat

CrIS\_VIIRS\_MOD\_HEI.dat

CrIS\_VIIRS\_WGT.dat

CrIS\_VIIRS\_WGT\_HEI.dat

### Template Files

The system uses a number of template files. These are all static files that will only change with a new delivery of the system. They are never modified by the scripts and programs that use them. Scripts will only copy these files to a local directory or create soft links to them

#### CDL Template Files

These are template parameter files used for generating the NUCAPS SDR and NUCAPS EDR granule subsets. These files contain the lists of channels and footprints to be extracted for each type of subset. They also contain the variable lists, array sizes and array dimensions for each NetCDF output file. Each file can be converted into a NetCDF file using the *ncgen* NetCDF4 library utility. This file will have a complete header based on that of the CDL template, but contains no instrument data values, only fill (missing) values. These files are then populated with instrument data values by the subsetter code. There is a different template file for each type of subset.

The following NUCAPS CDL template files shown in Table 7‑7 are present in the current build:

Table 7‑7 NUCAPS CDL Files

|  |  |
| --- | --- |
| CDL Template Name | Description |
| nucaps\_all.cdl | A template for all fovs, 1317 channels (4 scans) |
| nucaps\_c0300\_allfovs.cdl | A template for all fovs, 399 channels (4 scans) |
| nucaps\_c0300\_allfovs\_pcs1b.cdl | A template for all fovs, 399 channels of RR, 1 band (4 scans) |
| nucaps\_c0300\_allfovs\_pcs3b.cdl | A template for all fovs, 399 channels of RR, 3 band (4 scans) |
| nucaps\_c0300\_centerfov.cdl | A template for center fov, 399 channels (4 scans) |
| nucaps\_c0300\_centerfov\_pcs1b.cdl | A template for center fov, 399 channels of RR, 1 band (4 scans) |
| nucaps\_c0300\_centerfov\_pcs3b.cdl | A template for center fov, 399 channels of RR, 3 band (4 scans) |
| nucaps\_c1317\_1scan.cdl | A template for all fovs, 1317 channels, 1 scans/granule (4 scans) |
| nucaps\_c1317\_2scan.cdl | A template for all fovs, 1317 channels, 2 scans/granule (4 scans) |
| nucaps\_ccr.cdll | A template for 1 fov, 1317 channels (4 scans) |
| nucaps\_ccr\_archive.cdl | A template for 1 fov, 1317 channels (4 scans). Unused variable names are deleted and current ones are standardized for archive. Also contains static global attributes as metadata. |
| nucaps\_l2.cdl | A template for 1 fov for level 2 EDR profile data (4 scans) Also contains static global attributes as metadata. |
| cris\_olr.nc | A template for the OLR output netCDF file. Also contains static global attributes as metadata. |
| All files contain radiances unless otherwise indicated with an RR which stands for Reconstructed Radiances from principal components. | |

#### Matchup Template

This is a static text file containing a list of locations to which CrIS observations will be collocated. In the current build of the system these are only radiosonde launch sites and times. This matchup file name is *radiosonde\_machup\_template.*

The file is an ASCII file containing the following fields for each matchup location: latitude, longitude, year, month, day, hour, minute, distance tolerance, and time tolerance. The distance tolerance is the distance (in kilometers) within which CrIS observations will be considered for a collocation. The time tolerance is the time (plus or minus minutes) around which CrIS observations are considered for a collocation. The entire template list covers a 30 hour range of data such that it includes 24 hours from the target day plus the three hours that overlap at the end of the previous day and the beginning of the following day. The date information in the template consists of only text tags. A script called *Create\_Matchup\_List.pl* generates a matchup file for a given day based on this template by modifying the date tags to insert the desired processing date’s data fields.

## Reference Data Files

The retrieval namelist files are already identified in Table 7‑4 and Table 7‑6. These are 7 input parameter files that can be conceived of a both ancillary and reference data. These are ASCII text Fortran namelist files. These files contain the names of all the retrieval input files identified Table 7‑6 that are to be used in the operational configuration. Flags can be set to vary the retrieval processing steps, the types of input system files used, or to change the verbosity of the diagnostic output. None of these settings should ever be changed in the operational environment or it may result in degradation of quality or latency. These settings are available only to allow runtime flexibility in the offline science development environment at STAR.

## Intermediate Data Files

The full intermediate data files list is shown in Table 7‑2~ Table 7‑5. Additional information about these files is provided in this section.



### NetCDF Files

The NetCDF files are mainly intermediate files. These files are generated for NUCAPS SDR and CCR. There are four main programs that generate NetCDF. The files that each of these programs produce are discussed below.

The SDR NetCDF files are generated using main\_nucaps\_preprocessor from input data files. This program is part of the Preprocessor unit. Only one NetCDF file is produced for each granule with the following name structure.

NUCAPS\_ALL\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time HHMMSSS (hour, minute, second, and tenth of a second)

*${Granule\_End}* = the 7-digit granule end time HHMMSSS (hour, minute, second, and tenth of a second)

The OLR netCDF granule files are generated by main\_cris\_olr.f90

NUCAPS-OLR\_v1r0\_npp\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}..nc

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time HHMMSSS (hour, minute, second, and tenth of a second)

*${Granule\_End}* = the 7-digit granule end time HHMMSSS (hour, minute, second, and tenth of a second)

*${Version}* = version number (v06-08-10)

*${Data\_Start}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_End}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_Creation}* = the 14-digit product file creation time YYYYMMDDHHMMSS (4-digit year, 2-digit month, day, hour, minute, second)

The other SDR NetCDF files are generated using main\_nucaps\_subset from the previous SDR NetCDF files. This program is part of the Subsetter unit. The NetCDF files are produced for each granule with the following name structure.

NUCAPS\_C0300\_ALLFOVS\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C0300\_ALLFOVS\_PCS1B\_ALLFOVS\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C0300\_ALLFOVS\_PCS3B\_ALLFOVS\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C0300\_CENTERFOV\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C0300\_CENTERFOV\_PCS1B\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C0300\_CENTERFOV\_PCS3B\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_C1317\_1SCAN\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time *HHMMSSS* (hour, minute, second, and tenth of a second)

*${Granule\_End}* = the 7-digit granule end time *HHMMSSS* (hour, minute, second, and tenth of a second)

The EDR and CCR NetCDF granule files are generated from *main\_nucaps\_l2\_to\_netcdf* and *main\_nucaps\_ccr\_to\_netcdf*, respectively. The input files are the native binary format retrieval output files. These two programs are part of the Retrieval Unit. The following NetCDF intermediate files are produced for each granule. These files may be NUCAPS output files too.

NUCAPS\_CCR\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.nc

NUCAPS\_CCR\_AR\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.nc

NUCAPS\_EDR\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.nc

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time HHMMSSS (hour, minute, second, and tenth of a second)

*${Granule\_End}* = the 7-digit granule end time HHMMSSS (hour, minute, second, and tenth of a second)

*${Version}* = version number (v06-08-10)

*${Data\_Start}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_End}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_Creation}* = the 14-digit product file creation time YYYYMMDDHHMMSS (4-digit year, 2-digit month, day, hour, minute, second)

The EDR file,

*NUCAPS\_EDR\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.nc*, is sent to the OSDPD monitoring machine (nucaps.nesdis.noaa.gov) for use in their near real time monitoring. These files are about 3 MB each.

### Retrieval Input/Output Files

The retrieval input and output files are purely intermediate files and are therefore not distributed to any user. It is worth being aware of their names and locations for diagnostic purposes.

There are four input files required by the retrieval. These are all binary sequential access files for NUCAPS SDR and GFS surface pressure. The files have the following name structures:

atms\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.binary

cris\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.binary

avntest\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}.bin

avntest\_${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}b.bin

These last two files are named differently, but have identical contents for NUCAPS. The retrieval requires that they have these slightly different name structures. It’s worth noting here that this retrieval algorithm is designed to work with other instruments from other satellites as well so there some aspects of the implementation that may seem odd to the uninformed user.

There are a number of retrieval output files which are generated in the same directory:

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.ccr

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.f61

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.f69

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.fg

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.mit

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.ret

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_nucaps\_retrieval.bin

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_nucaps\_retrieval.f70

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_nucaps\_retrieval.f89

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_nucaps\_retrieval.f95

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_nucaps\_retrieval.out

where:

*${Year}* = the 4-digit year

*${Month}* = the 2-digit month

*${Day}* = the 2-digit day of month

*${Granule\_Start}* = the 7-digit granule start time HHMMSSS (hour, minute, second, and tenth of a second)

*${Granule\_End}* = the 7-digit granule end time HHMMSSS (hour, minute, second, and tenth of a second)

The retrieval profile output is located in:

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.ret

This file contains retrievals of temperature, moisture, and trace gases. The CCR file is:

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.ccr

These radiances are identical to the original CrIS radiance except the effects of clouds have been removed.

The MIT file contains information from the microwave portion of the retrieval.

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.mit

The first guess file contains the first guess fields.

${Year}${Month}${Day}\_${Granule\_Start}\_${Granule\_End}\_bin\_0002.fg

All the other files are diagnostic files and are therefore not used for any products downstream. These files are kept because they could be used by the NUCAPS science development team to determine the source of problems with the retrieval in the event that there are any. These files are all under 1 MB in size and are therefore not a burden to retain temporarily.

## Output Data Files

The full output data files list is shown in Table 7‑2~ Table 7‑5. Additional details are provided in this section.

### Global Grids

These are direct access daily binary files in a GRADS readable format such that each binary record consists of a single grid’s worth of data.

Below are listed the global grid files. The date information in the file names is defined as:

*${Version}* = version number (v1r0, v06-08-10)

*${Data\_Start}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_End}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_Creation}* = the 14-digit product file creation time YYYYMMDDHHMMSS (4-digit year, 2-digit month, day, hour, minute, second)

0.5 X 2.0 degree ascending and descending SDR global grids:

NUCAPS-GG-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

3.0 X 3.0 degree ascending and descending full SDR global grids:

NUCAPS-FG-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-FG-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending 1-band PC global grids:

NUCAPS-GG-PCS1B-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-PCS1B-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

3.0 X 3.0 degree ascending and descending 1-band PC global grids:

NUCAPS-FG-PCS1B-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-FG-PCS1B-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending 3-band PC global grids:

NUCAPS-GG-PCS3B-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-PCS3B-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending EDR global grids:

NUCAPS-GG-EDR-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-EDR-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending CCR global grids:

NUCAPS-GG-CCR-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-CCR-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending OLR global grids:

NUCAPS-GG-OLR-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-OLR-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

0.5 X 2.0 degree ascending and descending GFS forecast global grids:

NUCAPS-GG-GFS-GRIDS-ASC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

NUCAPS-GG-GFS-GRIDS-DSC\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

### Global Binaries

These are direct access binary files. Each record contains all the data for a single CrIS FOV of data. Each file contains all the data for a given day.

Below are listed the global binary files. The date information in the file names is defined as:

*${Year}* = the 4-digit year

*${Month}* = 2-digit month

*${Day}* = 2-digit day of month

Global binary file for 1317 channels with the all CrIS FOVs from each FOR:

nucaps\_${Year}${Month}${Day}.binary

Global binary file containing only the locations and times of all FOVs from each FOR:

nucaps\_${Year}${Month}${Day}\_locs.binary

Global binary file for all EDR profiles and all CrIS FOR:

nucaps\_l2\_${Year}${Month}${Day}.binary

Global binary file containing only the locations and times of all FOVs from each FOR:

nucaps\_l2\_${Year}${Month}${Day}\_locs.binary

These two sets of global binary files for NUCAPS SDR, and EDR are generated from the full resolution NetCDF files. These are required downstream as inputs to the Global Matchups unit. Each of these global binary files contains exactly the same number of records. Each record, in turn, represents all the data for a single field of regard. Therefore, any given record number (footprint) in one file corresponds to the same record number (footprint) in any other file. To maintain this correspondence, missing scans/footprints, if they occur, are added as placeholders. This correspondence is necessary for the functionality of the matchups.

### Global Matchups

There are two matchup output files. Each is a direct access binary file containing all the instrument data on a single instrument field of regard at a given radiosonde launch time and location. Each file contains all the data for the given day plus the last three hours of the previous day and the first three hours of the following day.

Below are listed the matchup files. The date information in the file names is defined as:

*${Version}* = version number (v1r0, v06-08-10)

*${Data\_Start}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_End}* = the 15-digit granule start time YYYYMMDDHHMMSSS (4-digit year, 2-digit month, day, hour, minute, second, and tenth of a second)

*${Data\_Creation}* = the 14-digit product file creation time YYYYMMDDHHMMSS (4-digit year, 2-digit month, day, hour, minute, second)

The binary global matchup file for 1317 channels with all CrIS FOVs from each FOR:

NUCAPS-SDR-GLOBAL-MATCHUPS\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

The ASCII text matchup list file associated with the NUCAPS SDR radiosonde matchups is located in the same directory and name, but with the .txt extension:

NUCAPS-SDR-GLOBAL-MATCHUPS\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.txt

The global matchup binary file for NUCAPS EDR is:

NUCAPS-EDR-GLOBAL-MATCHUPS\_${Verson}\_npp\_s${Data\_Start}\_e${Data\_End}\_c${Data\_Creation}.bin

### EDR Files

These two netCDF4 granule files in this section are to be externally distributed. The NUCAPS EDR file contains the 100-level profiles of temperature, moisture, and trace gasses

NUCAPS-EDR\_${Version}\_npp\_s${Data\_Start}\_e${Date\_End}\_c${Data\_Creation}.nc

The NUCAPS CCR file is the Cloud-Cleared Radiance (CCR) file produced by the retrieval process. These files contain the 1317 channel set at the resolution of the retrieval (40km at nadir).

NUCAPS-CCR-AR\_${Verson}\_npp\_s${Data\_Start}\_e${Date\_End}\_c${Data\_Creation}.nc

The NUCAPS OLR file is the Outgoing Longwave Radiance (OLR) file produced by the OLR code.

NUCAPS-OLR\_v${Verson}r{Release}\_npp\_s${Data\_Start}\_e${Date\_End}\_c${Data\_Creation}.nc

### Quality Monitoring Files

The files described in this section are generated specifically for OSPO quality monitoring. The PCS granule monitoring file contains statistics on the reconstruction scores of the radiances from all 27 CrIS detectors and has the following file name pattern:

NUCAPS-PCS-MONITORING\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt

The EDR granule monitoring file contains retrieval statistics for a given granule. It is the retrieval \*.out file renamed to adhere to the NDE naming convention. It has the following file name pattern:

NUCAPS-EDR-MONITORING\_ v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.txt

## Archive Data Files

Two NUCAPS files have been proposed for archive. These files are already described in section 7.6.4. In addition, these are already identified in Table 7‑4 as:

The NUCAPS EDR file which contains the retrieved profiles:

NUCAPS\_EDR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc

The NUCAPS CCR file which contains the cloud-cleared radiances:

NUCAPS\_CCR\_AR\_v1r0\_npp\_s???????????????\_e???????????????\_c??????????????.nc

These are both EDR products in netCDF4 format. Even though CCR is radiance, it is produced by the retrieval code. Details of the file contents are identified in the NUCAPS EUM (NESDIS/STAR, 2011). The file names adhere to the NDE naming convention for output files. The Submission Agreement is being developed between STAR and CLASS. Metadata have been developed for these files. They are inserted into the netCDF4 headers as global attributes and are CF-1.5 compliant.

# List of References

NDE (2007), NDE Critical Design Review (NDE\_CDR-20070918).

NDE (2008a), NDE Product Generation Application Interface, Version 0.4.

NDE (2011) Standards for Algorithm Delivery and Integration Using Delivered Algorithm Packages (DAPs), Version 1.3

NESDIS/STAR (2008a), NUCAPS Interface Control Document, Version 1.0.

NPOESS(2009), [NPOESS Common Data Format Control Book](http://www.star.nesdis.noaa.gov/smcd/spb/iosspdt/qadocs/NUCAPS_CDR/D34862-03_B_CDFCB-X_Volume_III.doc), Version D.

NESDIS/STAR (2009), NUCAPS Algorithm Theoretical Basis Document, Version 1.0.

NESDIS/STAR (2010), NUCAPS Software Architecture Document, Version 1.0.

NESDIS/STAR (2010), [NUCAPS Test Readiness Document (TRD)](http://www.star.nesdis.noaa.gov/smcd/spb/iosspdt/qadocs/NUCAPS_TRR/NUCAPS_TRD_Final.pptx)

NESDIS/STAR (2011) NUCAPS External Users Manual, Version 2.0.

NESDIS/STAR (2011) NUCAPS Requirements Allocation Document, Version 1.0.