NOAA NESDIS

CENTER for SATELLITE APPLICATIONS and RESEARCH

NDE Vegetation Products System (NVPS) Maintenance Manual

Version 1.1

Title: NDE VEGETATION PRODUCTS System Maintenance manual VERSION 1.1

AUTHORS:

Mingshi Chen (IMSG)

Zhangyan Jiang (IMSG)

Feng Zhao (IMSG)

Yuxiang He (IMSG)

Ivan Csiszar (STAR)

Yunyue Yu (STAR)

APPROVAL SIGNATURES:

Yunyue Yu November 28, 2018

NVPS Project Lead Date

DOCUMENT HISTORY

DOCUMENT REVISION LOG

The Document Revision Log identifies a series of revisions to the document since the baseline release. Please refer to the above page for version number information.

DOCUMENT TITLE: NVPS System Maintenance Manual			
DOCUMENT CHANGE HISTORY			
Revision No.	**Date**	**Revision Originator Project Group**	**CCR Approval # and Date**
1.0	08/21/2017	Created by Mingshi Chen	
1.1	11/19/2018	Created by Feng Zhao and Mingshi Chen	

LIST OF CHANGES

Significant alterations made to this document are annotated in the List of Changes table.

DOCUMENT TITLE: NVPS System Maintenance Manual					
LIST OF CHANGE-AFFECTED PAGES/SECTIONS/APPENDICES					
Version Number	**Date**	**Changed By**	**Page**	**Section**	**Description of Change(s)**
1.0	08/21/2017	Mingshi Chen	All	All	First release of the NVPS SMM.
1.1	11/19/2018	Mingshi Chen and Feng Zhao	All	All	Revision for the NOAA-20 Beta Release Review

---

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

IMSG	I.M. Systems Group
ATBD	Algorithm Theoretical Basis Document
CLASS	Comprehensive Large Array-data Stewardship System
DAP	Delivered Algorithm Package
DDS	Data Distribution Server
DHS	Data Handling System
GAG	NVPS Aggregator Unit
GCL	NVPS Calculator Unit
GRD	Surface Reflectance Gridder Unit
GRIB	Gridded Binary format
NVPS	NDE Vegetation Products System
HDF	Hierarchical Data Format
IMSG	I. M. Systems Group
IVISR	VIIRS Surface Reflectance RIP file
JPSS	Joint Polar Satellite System
NCEP	National Centers for Environmental Prediction
NDE	NPOESS Data Exploitation
NESDIS	National Environmental Satellite, Data, and Information Service
NetCDF	Network Common Data Form

NOAA	National Oceanic and Atmospheric Administration
S-NPP	Suomi National Polar-orbiting Partnership
RAD	Requirements Allocation Document
RAS	Requirements Allocation Sheet
QA	Quality Assurance
SAVI	Soil Adjusted Vegetation Index
SRC	Surface Reflectance Compositer Unit
STAR	Center for Satellite Applications and Research
SVI	Smooth EVI Unit
SWA	Software Architecture Document
TBD	To Be Determined
TGM	Tile-Granule Mapper Unit
TIFF	Tagged Image File Format
VIF	EVI Filter Sub-Unit
VIIRS	Visible Infrared Imager Radiometer Suite

EXECUTIVE SUMMARY

The NOAA Unique NDE Vegetation Products Processing System (NVPS) is developed to generate a consistent set of global and regional gridded vegetation products from S-NPP and JPSS VIIRS observations for initializing environmental models and monitoring land use and land cover change. NVPS consists of both Green Vegetation Fraction (GVF) and Vegetation Index (VI) subsystems. GVF subsystem is operationally running within the NPOESS Data Exploitation (NDE) production environment to generate weekly GVF data at both global (pixel size of 0.036°) scale and regional (pixel size of 0.009°) scale. VI subsystem is recently embedded into NVPS to generate the daily, weekly and bi-weekly global (0.036°) and regional (0.009°) gridded VI products and quality assurance (QA) flags with thematic information about the quality of the VI products. NVPS VI products include the Top of the Atmosphere (TOA) Normalized Difference Vegetation Index (NDVI), Top of the Canopy (TOC) NDVI as well as the TOC Enhanced Vegetation Index (TOC EVI). NVPS software architecture is displayed in an external interface, a context layer and two subsystem data flow diagrams.

NVPS is being delivered to the Office of Satellite and Product Operations (OSPO) through NDE to produce NVPS products on a Linux platform. NVPS is identified as a critical mission and therefore a 24×7 service maintenance level required. Return to service within 2 hours is required. Requirements for the NVPS products are described in the NVPS Requirements Allocation Document (RAD) (NESDIS/STAR, 2017) which is available in the NVPS project artifact repository.

The NVPS product team consists of the following team members:

Table 1. NVPS Product Team.

Team Member	Organization	Role	Contact Information
Yunyue Yu	STAR	STAR Product Leader	5830 University Research Court College Park, MD. 20740

			Phone: 301-683-2566 Email: yunyue.yu@noaa.gov
Ivan Csiszar	STAR	STAR Product Leader	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3583 Email: ivan.csiszar@noaa.gov
Mingshi Chen	STAR	Developer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-0518 Email:mingshi.chen@noaa.gov
Zhangyan Jiang	STAR	Developer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3528 Email:zhangyan.jiang@noaa.gov
Feng Zhao	STAR	Developer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3528 Email:feng.zhao@noaa.gov
Yuxiang He	STAR	Developer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3528 Email:yuxiang.he@noaa.gov

Hanjun Ding	STAR	Product Area Lead (PAL)	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3243 Email: hanjun.ding@noaa.gov
Yufeng Zhu	OSPO	Primary Maintenance Programmer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3276 Email: Yufeng.Zhu@noaa.gov
William Pennoyer	OSPO	Secondary Maintenance Programmer	5830 University Research Court College Park, MD. 20740 Phone: 301-683-3253 Email:William.Pennoyer@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 System (INS), Product Generation System (PGS), Product Distribution System (PDS), and Monitoring subsystems, and is physically spread across numerous server, 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 Red hat 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 are in a transition to run on Linux processing nodes, and more Linux processing capability will be added in the future. The NVPS algorithm is running on the Linux processing nodes.

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

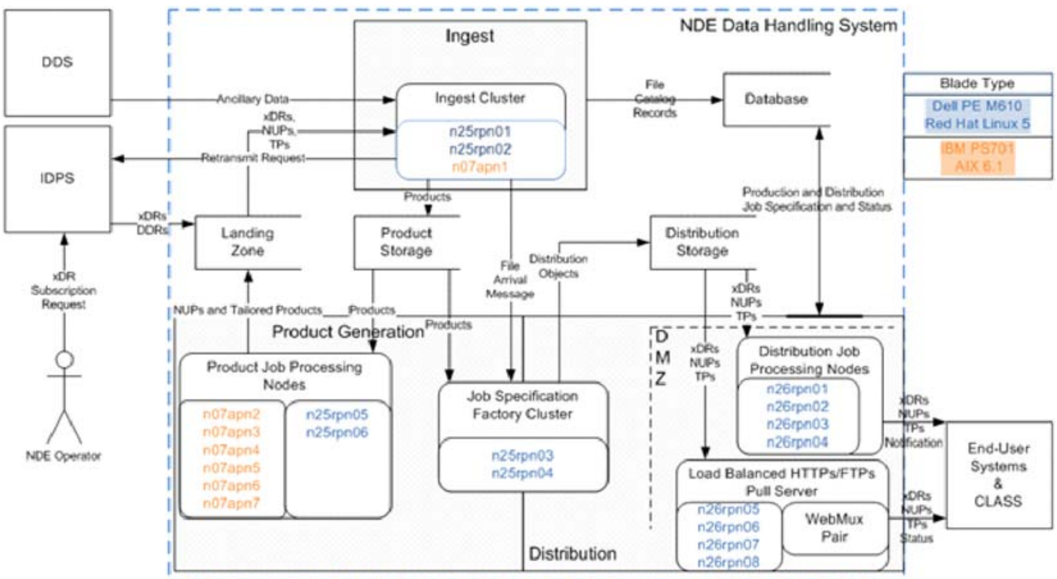


Figure 1. NDE Data Flow Diagram and Physical Layout

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

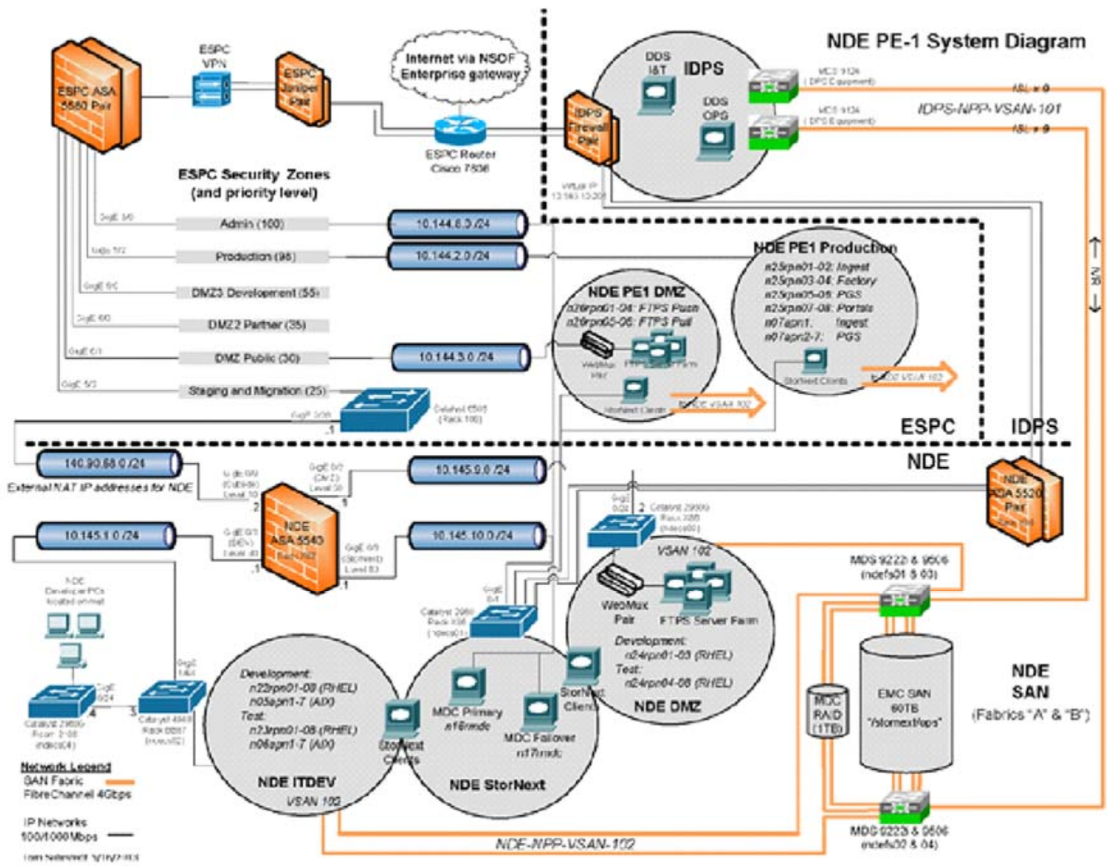


Figure 2. NDE System Diagram

All low-level processing codes of NVPS are based on C++ programming language while high-level codes are in the format of BASH scripts. Low-level C++ codes provide the core class and functionality of each processing unit. High-level Bash scripts are developed for each processing unit to invoke the low-level executables and an overall driver script in BASH integrates all the software units to work as a system.

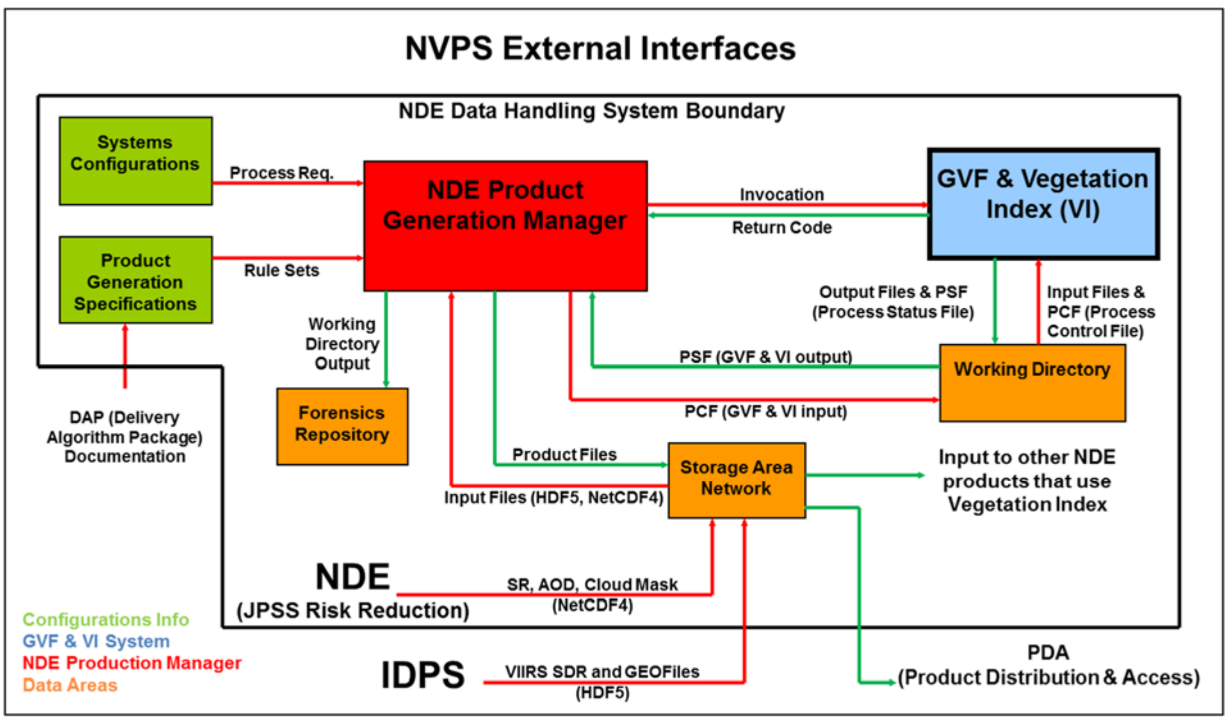


Figure 3. NVPS External Interfaces

For information about the NVPS algorithm, see the NVPS Algorithm Theoretical Basis Document (ATBD).

The main VIIRS Level 1 input data are received from IDPS (HDF5 format) and NDE (NetCDF4 format). The NVPS products are currently distributed to users via NDE distribution server. Meanwhile, users can also find the archived products in CLASS.

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. 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 products 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) should be contacted for any data accessibility and data distribution problems.

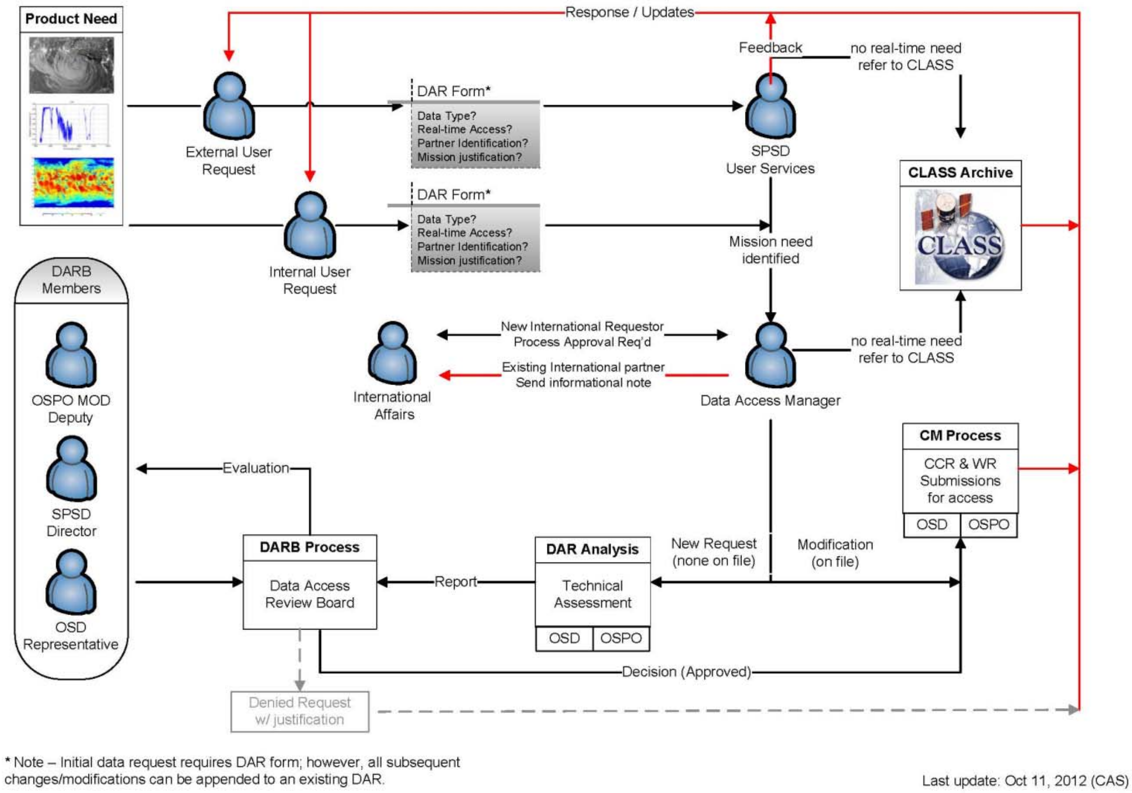


Figure 4. SEQ Figure * ARABIC 4. NDE Data Access Process

1. INTRODUCTION

1.1. Product Overview

The NOAA Unique NVPS processing system is developed for generating the current operational VIIRS Green Vegetation Fraction (GVF) and the new gridded Vegetation Index (VI) products. The VI products include the Top of the Atmosphere (TOA) Normalized Difference Vegetation Index (NDVI), Top of the Canopy (TOC) NDVI as well as the TOC Enhanced Vegetation Index (TOC EVI). Meanwhile, quality assurances (QA) of VI products are generated to consider impact of various environmental factors (e.g., cloudy, sun glint, aerosol). These VI products are produced at temporal scales of daily, weekly and bi-weekly at spatial resolutions with sizes of of 4-km (0.036°, global scale) and 1-km (0.009°, regional scale). GVF is only derived from the TOC EVI at the same spatial scales as VI and at temporal scale of weekly. The weekly and bi-weekly composited VI products and weekly GVF are generated in rolling way every day. This system ensures sufficient continuity with the current AVHRR-based system, and also introduces the necessary improvements to meet user requirements. The system is designed to run within the NPOESS Data Exploitation (NDE) system delivered to the Office of Satellite and Product Operations (OSPO) as an NDE Delivered Algorithm Package (DAP). The output products are intended for operational and scientific users. NVPS production system produces NVPS products on a Linux platform.

1.2. Algorithm Overview

The NVPS includes two sets of retrieval algorithms: GVF algorithm and VI algorithm.

GVF retrieval algorithms require the following inputs: daily granule TOC VIIRS red (I1), TOC VIIRS near-infrared (I2), and TOC VIIRS blue (M3) stored in NetCDF4 format, and ancillary geolocation and water mask stored in h5 format. The GVF algorithm is carried out in the following steps: 1) using gelocation information, grid and tile scheme to build tile-granule relationship; 2) mapping all available daily granule observations (in bands I1, I2, and M3) to the native GVF geographic grid (0.003° plate carree projection) to produce a gridded daily surface reflectance map; 3) compositing the daily surface reflectance maps of the last 7 days (including the current day) to produce a weekly surface reflectance map using the MVA-SAVI compositing algorithm, which selects, at each GVF grid point (pixel), the observation with maximum view-angle adjusted SAVI value in the 7-day period. The 7-day compositing is conducted with a daily rolling weekly compositing scheme, which uses data on every day in the previous 7 days as input. Cloud mask information was used in compositing and saved for the selected pixels during compositing; 4) calculating EVI from the daily rolling weekly composited VIIRS surface reflectance data in bands I1, I2 and M3; 5) reducing high frequency noise in EVI by applying a smoothing filter on EVI of the previous 15 non-overlapping weeks; 6) calculating GVF by comparing the smoothed EVI against the global maximum and minimum EVI values assuming a linear relationship between EVI and GVF; 7) aggregating GVF from 0.003° to 0.009° and 0.036° resolution for output maps.

VI retrieval algorithms require inputs including daily granule TOA VIIRS red (I1), TOA VIIRS near-infrared (I2) reflectance bands stored in h5 format, TOC VIIRS red (I1), TOC VIIRS near-infrared (I2), and TOC VIIRS blue (M3), as well as quality assurance data (e.g., cloud mask, aerosol optical thickness (AOT)) stored in NetCDF4 format, and ancillary geolocation and water mask stored in h5 format. The VI algorithm is carried out in following steps: 1) using geolocation information, grid and tile scheme to build tile-granule relationship; 2) mapping all available daily granule input data to the native VI geographic grid (0.003° plate carree projection) to produce a gridded daily TOA and TOC reflectance maps with quality assurance information; 3) if calculating daily VI, then skipping this step. Compositing the daily TOA and TOC reflectance maps of the last 7 and 16 days (including the current day) to produce a weekly and bi-weekly TOA and TOC reflectance maps using the MVA-SAVI compositing algorithm, which selects, at each VI grid point (pixel), the observation with maximum view-angle adjusted SAVI value in the 7-day or 16-day period. The 7-day and 16-day compositing is conducted with a daily rolling weekly and bi-weekly compositing scheme, which uses data on every day in the previous 7 and 16 days as inputs. Cloud mask information used in compositing and saved for the selected pixels in compositing; 4) aggregating daily, weekly and bi-weekly TOA and TOC reflectance maps from 0.003° to 0.009° and 0.036° resolution for output maps;

5) using NDVI and EVI schemes to calculate daily, weekly, and bi-weekly TOA NDVI, TOC NDVI and TOC EVI from the daily reflectance maps in step 2 and weekly and bi-weekly composited reflectance maps in step 3, respectively; 6) using spatial aggregation schemes to assess quality assurances of VI products mentioned in the step 5.

For detailed information about the NVPS algorithms, see the NVPS Algorithm Theoretical Basis Document (NESDIS/STAR, 2017).

1.3. Interfaces Overview

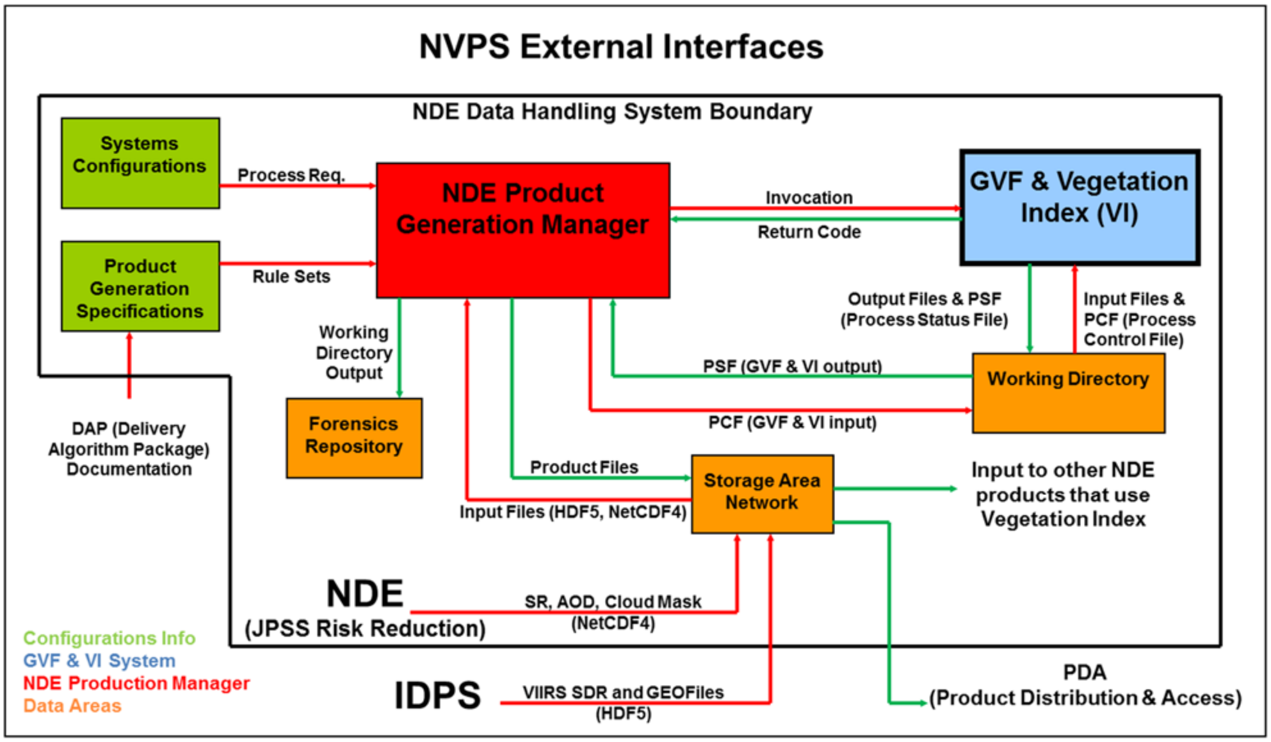


Figure 5. NVPS External Interfaces.

2. HARDWARE

2.1. Hardware Description

The hardware to support the NDE Production Environment-1 environment includes computing platforms that support AIX and Red Hat Linux as well as various networking equipments. Some components of the current PE-1 environment are shared resources while other components are PE-1 dedicated resources.

A description of the hardware required for the NDE PE-1 environment requires a basic understanding of the interconnectivity between NDE and the ESPC zones, IDPS, and storage. The figure below illustrates the NDE PE-1 connectivity.

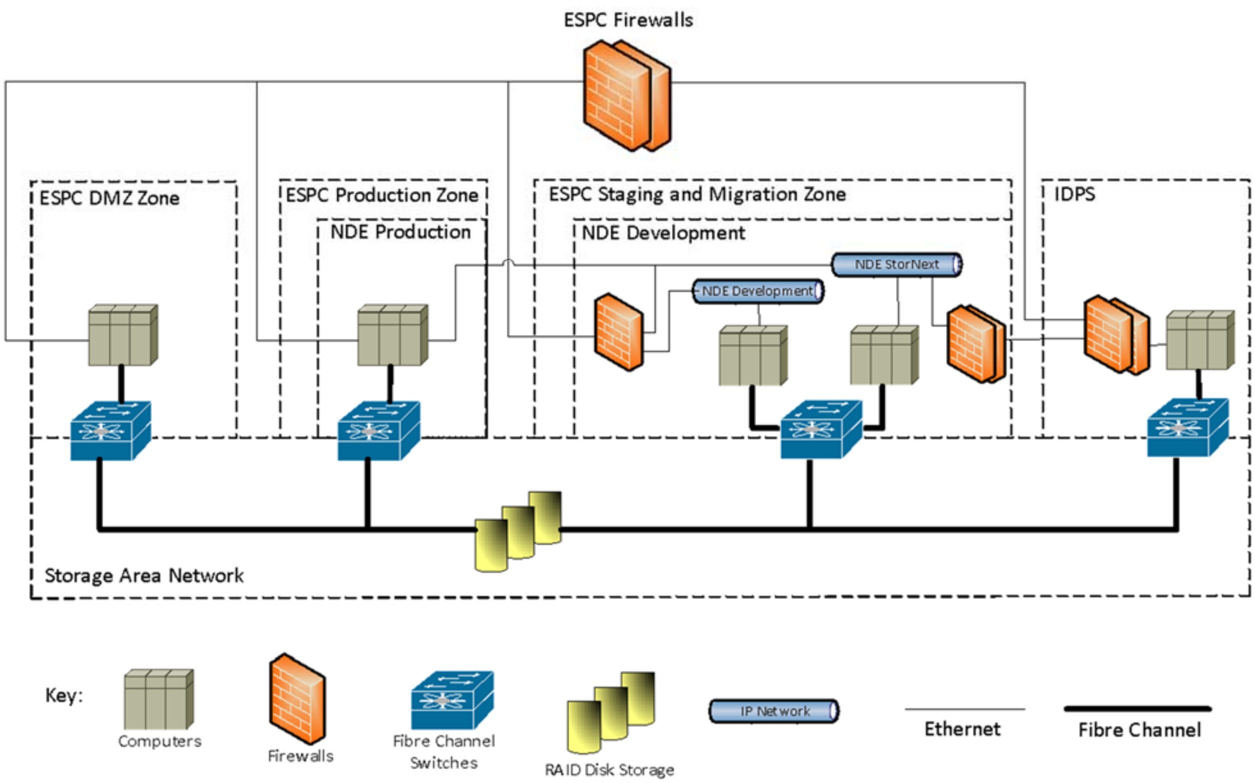


Figure 6. A High-level Logical Diagram of the Connectivity Used to Describe NDE PE1 Hardware

The NDE Data Handling System (DHS) consists of Ingest System (INS), Product Generation System (PGS), Product Distribution System (PDS), and Monitoring subsystems; and is physically distributed 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. Summary of PE-1 Hardware.

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
(2)EMC AX4-5F Storage Array	Local Storage for R610 servers
Dell M1000E BladeCenter Chassis	
(8) Dell M610 Blade Servers	Ingest, Tailoring,
(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	FTPS/HTTPS 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 file system 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]

The following table maps the hardware items listed above to their physical interface (see Figure 6).

Table 3. Mapping of PE-1 Hardware to Physical Interface.

.

Device	Support Group	Physical Interfaces			
		D	P	M	S
IBM BladeCenter H Chassis	ESPC		✓	✓	✓
IBM DS3512 Storage Array	ESPC		✓		✓
IBM EXP3512 Expansion Array	ESPC				
(2) Dell PowerEdge R610 Servers	ESPC		✓		

Dell M1000E BladeCenter Chassis	ESPC	✓	✓	✓	✓
EMC AX4-5F Storage Array	ESPC		✓		✓
(2) CAI Networks 690PG WebMuxes	ESPC	✓			
EMC Storage Array [ESPC controlled]	ESPC				✓
(2) Cisco MDS 9506 Fibre Channel Switches	NDE			✓	✓
(2) Cisco MDS 9222 Fibre Channel Switches	NDE			✓	✓
(2) Dell PowerEdge 2950 Servers	NDE			✓	✓
LSI Storage Array	NDE			✓	✓
(2) Cisco ASA 5500 Series Firewalls	NDE			✓	
Cisco Catalyst 2960 Ethernet switch	NDE				✓

Support Group = responsible administrative group after delivery/turnover of PE-1 to ESPC

Physical Interfaces = environments on which unit has physical interfaces

D=DMZ

P=Production

M=StorNext Ethernet metadata network

S=SAN Fibre Channel network

2.1.1. Storage Area Network

The most widely shared resource is the storage network used across the entire environment shown in Figure 6: A high-level logical diagram of the connectivity is used to describe NDE PE1 hardware. The StorNext SAN file system form Quantum is used to manage this common storage across platforms. An Ethernet network carries the metadata for StorNext file systems, and is used as the front-end for all clients to access files on a StorNext file system.

The SAN itself is implemented on a Fibre Channel network. All hosts that require access to SAN storage must have integrated Host Bus Adapters (HBAs) that provide the necessary Fibre, Channel connectivity. NDE administers two Cisco MDS Fibre Channel switches on each of two fabrics for this network. EMC Symmetrix disk storage administered by ESPC is used for PE1. Though the StorNext file system that is used for PE1 is currently also used for testing within NDE, an redundant dedicated storage area is configured on ESPC's EMC Symmetrix for the testing environment to be PE-2.

The satellite data provided by IDPS is placed on the SAN. NDE production servers use the data to generate products which are also placed on the SAN and then supplied to external customers. The FTPS protocol is used to distribute data from the DMZ. Some customers choose to have data pushed to them while others pull data directly from the DMZ. External users are provided with accounts on NDE Distribution servers in the DMZ, and they can get access to the data available on the SAN when they log in. The FTPS protocol has been selected because it allows not only for an encrypted FTP control connection (for security) but also allows for an unencrypted data connection (for a more efficient transfer of data).

2.1.2. StorNext Metadata Controllers

Currently, there are two Red Hat Enterprise Linux Version 5 (RHEL5) servers on the NDE StorNext network that serve as a failover pair of Metadata Controllers for StorNext. These are rack-mounted servers that control the StorNext resources for all environments. A separate RAID chassis provides local storage for these servers.

2.1.3. Dedicated Resources

Much of the equipment for PE1 is dedicated and not shared for any other purposes. A general description of these resources is provided as follows. More detailed description can be found in the section.

2.1.4. Dell Blade Servers

The Dell BladeCenter chassis contains eight blade servers. Each blade server is a single board that is placed in one slot of the BladeCenter. These Dell blades use Intel chips and provide the computing resources for one logical host running RHEL. Different Linux servers are used for ingesting, tailoring and distribution. The necessary Ethernet interfaces and Fibre Channel HBAs for each server are also integrated into these boards.

Each server requires one Ethernet interface for its main PE-1 network (either production or DMZ), and another for the StorNext network used for metadata control information. The HBAs are required for the Fibre Channel connection to the storage.

There are two Ethernet switches inside the BladeCenter chassis that are used to connect the Blade Servers to the Ethernet switches for the PE1 and StorNext networks. The HBAs connect to pass-through devices in the BladeCenter chassis to the NDE Fibre Channel switches. The BladeCenter chassis also has redundant power supplies.

Some blades are located in a BladeCenter chassis in the DMZ zone, and some are in a BladeCenter chassis in the PROD zone. All of them are connected to the SAN Fibre Channel network.

2.1.5. Dell M1000e PowerEdge BladeCenter Enclosure

There are two Dell M1000e PowerEdge BladeCenter chassis for PE-1. A M1000e chassis can include up to 16 half-height blades (server modules), providing support for each blade with its power supplies, fan modules, Chassis Management

Controller (CMC) modules, and I/O modules for external network connectivity.

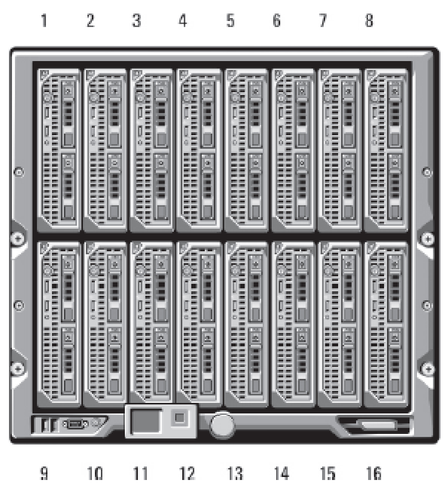


Figure 7. Each M1000e Chassis Contains Eight PowerEdge M610 Half-height Blade Servers

Each of the M610 blade server cards contains two Xeon 5660 series 6-core 2.80 GHz Intel processors with 36 GB of memory, two 300GB SAS disk and dual Ethernet and Fibre Channel interfaces. Connections to the external networks are provided through the cards shown below:

Table 4. External Network Cards

Location	Description	Function
Slot A1	Cisco Catalyst WS-CBS3130G-S Ethernet Switch	Ethernet switch for Production network
Slot A2	Cisco Catalyst WS-CBS3130G-S Ethernet Switch	Ethernet switch for StorNext network

Slot B1	Fibre Channel SAN Pass-Through Module	Fibre Channel Connectivity
Slot B2	Fibre Channel SAN Pass-Through Module	Fibre Channel Connectivity
Primary CMC Bay	Chassis Management Controller (CMC)	Primary CMC
Secondary CMC Bay	CMC	Secondary CMC

2.1.6. Dell M1000e PowerEdge BladeCenter Local Storage

In addition to the StorNext file system available on the SAN, an EMC AX-4 RAID subsystem contains redundant storage which is made available through the SAN to each of the eight RHEL5 hosts that are resident in the PROD Dell BladeCenter. The subsystem consists of an AX-4 chassis containing (2) controllers and (12) 600 GB SAS disk, and one chassis with (2) Standby Power Supplies that provide temporary battery backup to controllers in the event of a power failure.

2.1.7. Dell M1000e PowerEdge BladeCenter Enclosure

The Oracle database is hosted on a pair of Dell PowerEdge R610 rack-mounted servers. Each server has a single 3.33 Ghz X5680 Xeon processor. As configured for PE-1, each server also has:

96 GB memory

(2) 149 GB SSD disks

Four integrated 1 Gbps Ethernet NICs

(2) QLogic QLE2562 Fibre Channel HBAs

2.1.8. Dell M1000e PowerEdge BladeCenter Enclosure

Each of the Oracle servers has an EMC AX-4 RAID subsystem dedicated and directly attached to it. The subsystem consists of an AX-4 chassis containing (2) controller and (12) 600 GB SAS disks, and one chassis with (2) Standby Power Supplies that provide temporary battery backup to controllers in the event of a power failure.

2.1.9. IBM Blade Servers

The IBM BladeCenter chassis contains seven blade servers. Each blade server is a single board that is placed in one slot of the BladeCenter. These IBM blades use proprietary Power Chips from IBM and provide the computing resources for one logical host running IBM’s AIX operating system. The AIX servers are used for product generation and science algorithms. The necessary Ethernet interfaces and Fibre Channel HBAs for each server are also integrated on these boards.

Each server requires one Ethernet interface for the PE-1 network and another for the StorNext network used for metadata control information. The HBAs are required for the Fibre Channel connection to the storage.

There are two Ethernet switches inside the BladeCenter chassis that are used to connect the Blade Servers to the Ethernet switches for the PE1 and StorNext networks. The HBAs connect to Fibre Channel switches that are contained in I/O slots of the BladeCenter chassis for connectivity to the NDE Fibre Channel switches. The BladeCenter chassis also has redundant power supplies.

2.1.10. IBM BladeCenter H

The IBM BladeCenter H chassis contains Power supplies, switches and blades in 9U of rack space.

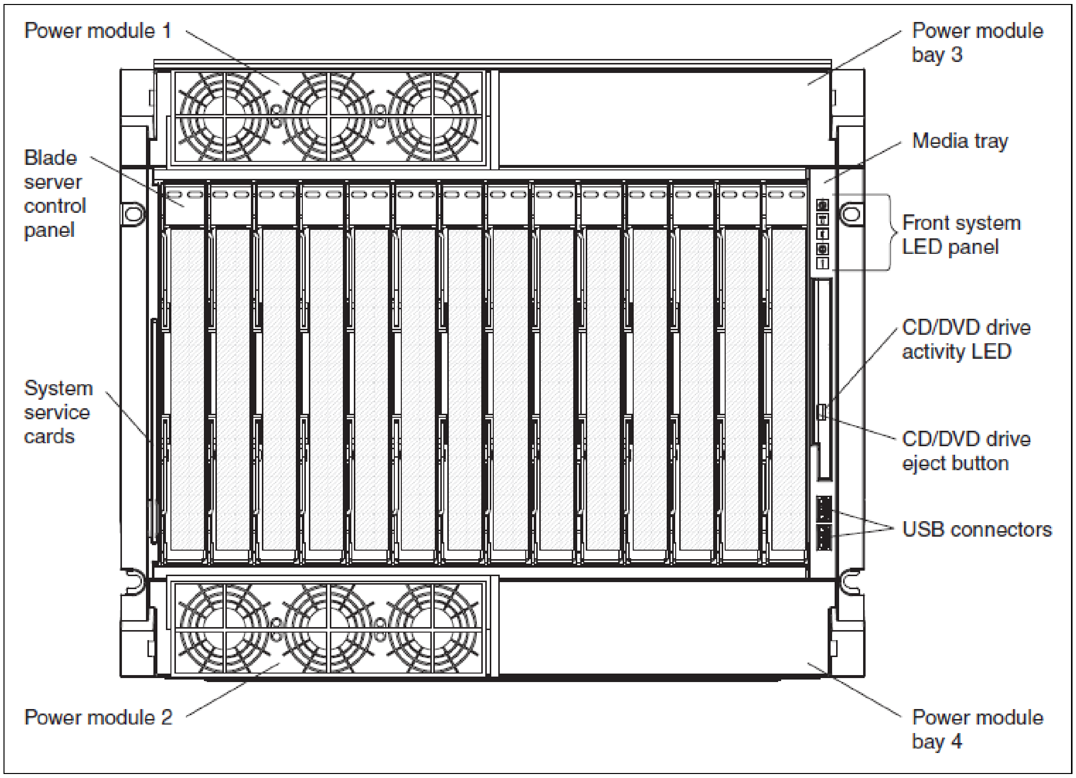


Figure 8. Front view of the BladeCenter H chassis, with 14 slots available for blades and shared media tray on the right

The chassis has 14 slots for blades, of which 7 are currently used to hold single-slot PS701 blades (model 8406-71Y), each of which blades having a single socket 8-core 3.0 GHz POWER7 64-bit processor. As configured for PE-1, each blade also contains:

64 GB memory

600 GB SAS disk

Two integrated 1 GB Ethernet NICs

One Emulex 8 GB Fibre Channel HBA Expansion card (CIOv) with 2 ports

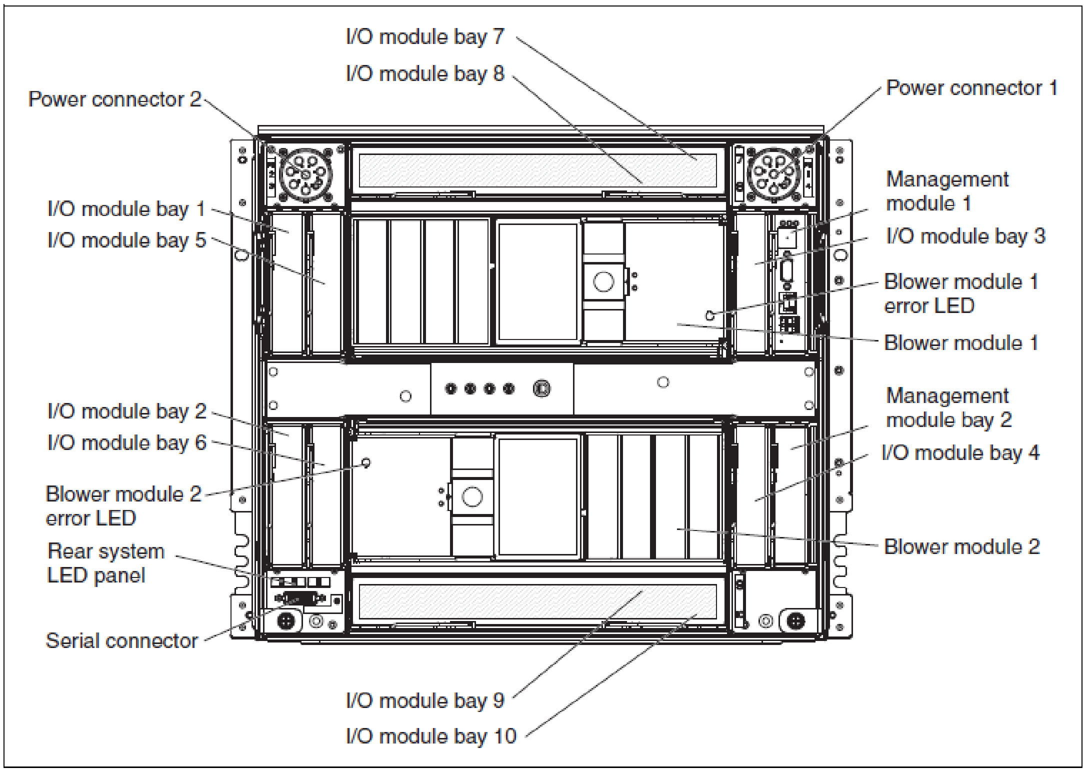


Figure 9. Rear View of Chassis

Table 5. PE-1 IBM BladeCenter H I/O Bays

Location	Destination	Function
I/O Bay 1	Cisco Catalyst Switch Module 3012	Ethernet switch for production network

I/O Bay 2	Cisco Catalyst Switch Module 3012	Ethernet switch for StorNext network
I/O Bay 3	Cisco 4 GB 10 port FC Switch Module	Fibre Channel Switch
I/O Bay 4	Cisco 4GB 10 port FC Switch Module	Fibre Channel Switch
Management Module Bay 1	Advanced Management Module (AMM)	Primary AMM
Management Module Bay 1	Advanced Management Module (AMM)	Standby AMM

2.1.11. IBM RAID Storage

There are two separate chassis that provide RAID storage for the AIX hosts, a main chassis with dual controllers, disk drives, and an expansion chassis containing just disk drives. Storage created on this RAID subsystem is assigned to each of the AIX servers. The RAID controllers are connected to the NDE Fibre Channel switches, and the AIX servers access the storage through the SAN.

2.1.12. IBM DS3512 Storage Array

In addition to the StorNext file system available on the SAN, an IBM DS3512 RAID subsystem contains redundant storage that is made available through the SAN to each of the seven AIX hosts that are resident in the IBM BladeCenter. The subsystem consists of one DS3512 chassis containing 2 controllers and 12 600 GB SAS disks, and one EXP3512 Expansion chassis containing an additional 12

600 GB SAS disks.

Figure 10. Front View of DS3512

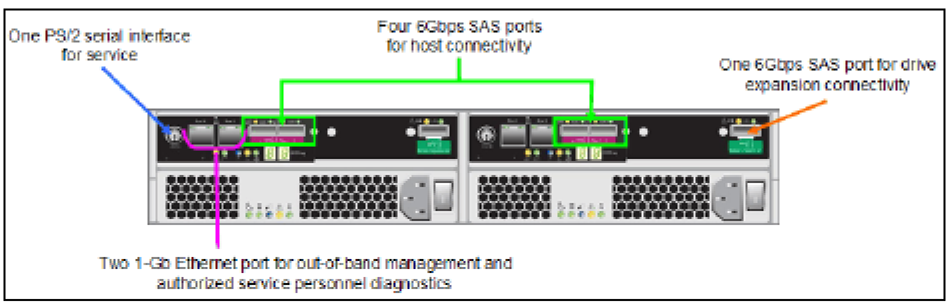


Figure 11. Rear View of DS3512

2.1.13. Oracle Database Servers

There are rack-mounted Dell servers that function as a redundant pair of Oracle database servers. Each runs RHEL5 on Intel processors.

2.1.14. EMC Storage for Dell Servers

There are three rack-mounted RAID storage subsystems from EMC that provide local storage for the Dell servers. One of these subsystems is dedicated as local storage for each of the Oracle servers, and is directly connected to the HBAs of these servers. The other subsystem is connected through the SAN and provides storage for all of the blade servers in the PROD Dell BladeCenter chassis.

2.1.15. WebMux Devices

Two WebMux devices are installed as a failover pair on the DMZ network. External users are able to connect to these devices to establish the FTPS control connection to NDE Distribution servers. Using a round-robin type of allocation algorithm, the WebMux passes the connect request to an FTPS server in the server rack. The FTPS server provides the clients with appropriate IP addresses and port information to establish a passive data connection. The external client will initiate a passive data connection directly to the distribution server to retrieve data.

There are two WebMux 690PG Load Balancer devices from AVANU, which are a front-end to the NDE distribution servers in the DMZ. These are configured as a redundant pair. The FTPS server farm is configured on round-robin and the HTTPS server farm is configured on persistent round-robin.

2.2. 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 AIX operating system using the standard set of XL compilers. In the future, all algorithms will be transitioned to Redhat Enterprise Linux with both the GNU and Intel set of 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) has 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.

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

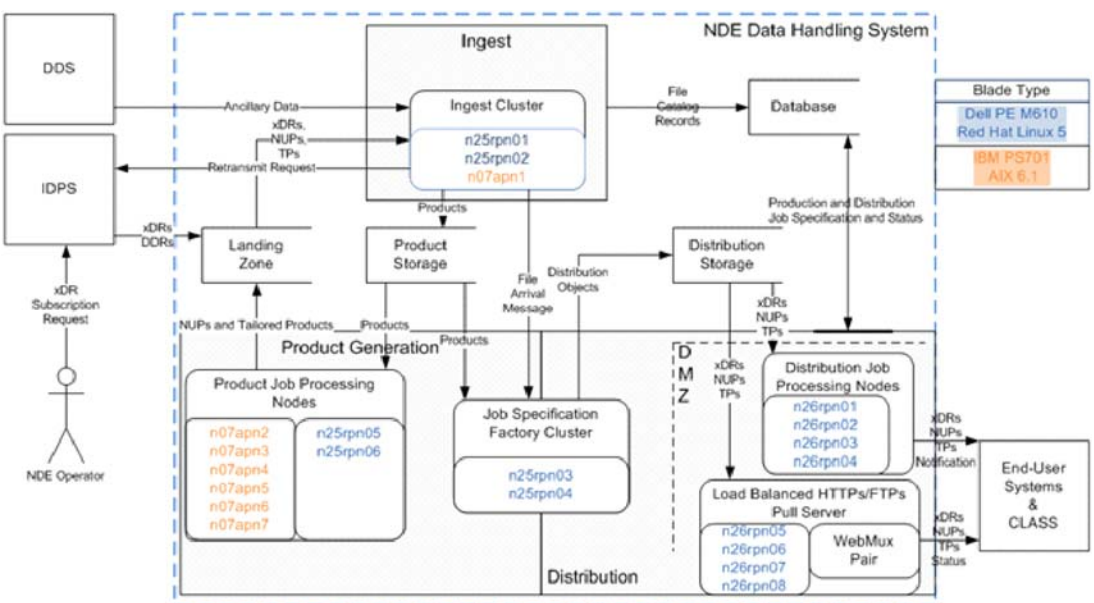


Figure 12. NDE Hardware Layout and Flow.

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

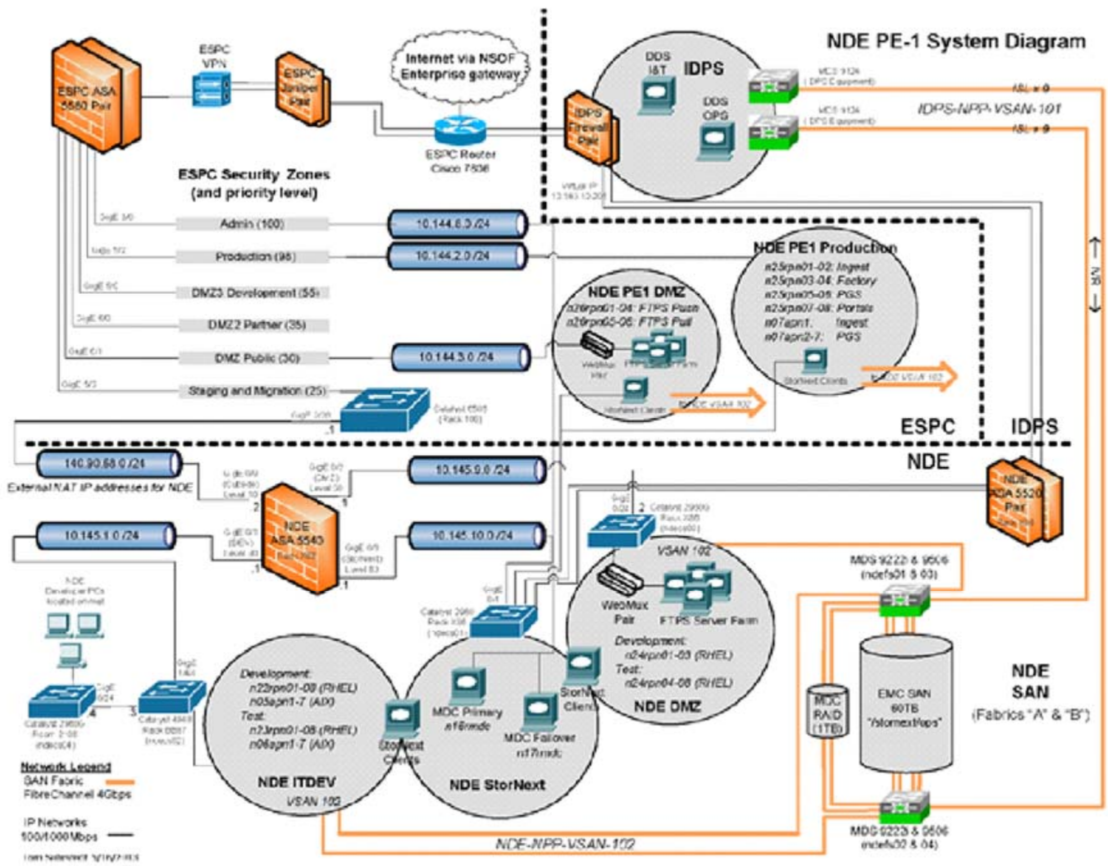


Figure 13. NDE System Diagram

2.3. System Requirements

2.3.1. Storage Requirements

Table 7 contains a list of storage requirements for all NVPS primary and ancillary input data, intermediate data, and output data. These requirements reflect the current build of NVPS.

Table 7. NVPS Storage Requirements

Storage Item	Size (GB)
Incoming VIIRS Data	1350
Intermediate Files	252
System Code (executable)	0.005
System scripts (interpreted)	0.0001
Ancillary file (climatology, water mask)	1.1
Data Products Files	26
Total	~1728

Tools, such as GNU 4.4.7 compiler and Bash utilities, are installed and managed by IT system administrators, and as such, are not included in the table. Similarly, tools, such as NetCDF4, HDF5, and szlib, are required by NVPS, but are not considered to be part of the delivery since they are an NDE managed and shared resources. In NDE operational environment, all incoming input and output data need a retention period of 5 days. The intermediate data include the gridded surface reflectance of the previous 6 and 15 days in weekly and biweekly cases, respectively, EVI of previous 14 weeks, and phase-one smoothed EVI of the previous 6 days.

2.3.2. Computer Resource Requirements

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

2.3.2.1. Processing Requirements

As identified from NVPS RAD, NVPS VIIRS VI and GVF products are available within 24 hours from satellite. The NVPS software was developed and tested on Linux. The most computational intensive units (e.g., GRD) require two CPUs for GVF and five CPUs for VI; the other units only require one CPU.

Table 8 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 NVPS granule. The maximum memory is defined as the maximum 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.

Table 8. NVPS VI & GVF System Processing Requirements.

Process for GVF Subsystem	**Max Memory**	**Time (hours)**
GRD	~ 3 GB	2
SRC	~ 3 GB	1
CVI	~ 1 GB	0.25
SVI	~ 2 GB	1
GCL	~ 1 GB	0.25

AGG	~ 3 GB	0.15
Process for VI Subsystem	**Max Memory**	**Time (hours)**
Daily GRD	~ 3 GB	1.40
Daily RAG, VIC & QAA	~ 4 GB	1.25
Weekly RCP	~ 3 GB	1:45
Weekly RAG, VIC & QAA	~ 4 GB	2
BiWeekly RCP	~ 4 GB	4:15
BiWeekly RAG, VIC & QAA	~ 3 GB	2.30

2.3.2.2. Libraries and Utilities

NVPS requires the following libraries and utilities:

GNU 4.4.7 compiler (e.g.,G++)

netCDF4 version 4.3.2

HDF5 version 1.8.13

TIFF 4.0.0 / GeoTIFF 1.4.0

uuid_gen

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

2.3.2.3. GNU g++ Compiler

Only the test machine requires the presence of a GNU g++ compiler. The production machine does not need a compiler because only the tested compiled code is necessary on the production machine. After the NVPSsystem has been successfully tested on the test machine, it is copied over to the production machine. However, only the compiled code is copied 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.

2.3.2.4. Bash Shell

The Bash shell scripts are used to invoke the NVPS executables compiled from C++ programming language. Bash shell is now the default shell on most Linux systems.

2.3.3. Communication Needs

Table 9 shows the data volume to be transferred to NDE while table 10 shows the data volume to be transferred out of NDE's data distribution server to external users. These two tables should be used as a guide for determining the disk space and bandwidth for the DHS.

Table 9. NVPS VI & GVF Input Data Sizing Per Provider

Product	Number of Files/Day	Size/Day	Provider
VIIRS geolocation HDF5	~500	48 GB	IDPS
VIIRS TOA Reflectance(I1) HDF5	~500	12 GB	IDPS
VIIRS TOA Reflectance (I2) HDF5	~500	13 GB	IDPS
VIIRS Surface Reflectance netCDF4	~500	102 GB	NDE

VIIRS Aerosol Optical Depth (AOD) netCDF4	~500	22 GB	NDE
VIIRS Cloud Mask netCDF4	~500	8 GB	NDE
Total	~3000	205 GB	IDPS & NDE

Table 10. NVPS VI & GVF Output Products Sizing Per User Per Day

Product	Number of Files/Day	Size/Day	User
NVPS VI Daily,Weekly,Biweekly Global 4km NetCDF	3	551 MB	BUFR toolkit
NVPS VI Daily,Weekly,Biweekly Regional 1km NetCDF	3	3.9 GB	BUFR toolkit
NVPS VI QA Monitoring	18	374 MB	OSPO
NVPS VI Daily,Weekly,Biweekly Global 4km GRIB2	3	551 MB	CLASS and NCEP
NVPS VI Daily,Weekly,Biweekly Regional 1km GRIB2	3	3.9 GB	CLASS and NCEP
Total NVPS VI	30	9.3 GB	

NVPS GVF Weekly Global 4km NetCDF	1	15 MB	BUFR toolkit
NVPS GVF Weekly Regional 1km NetCDF	1	60 MB	BUFR toolkit
NVPS GVF QA Monitoring	1	40 MB	OSPO
NVPS GVF Weekly Global 4km GRIB2	1	15 MB	NCEP
NVPS GVF Weekly Regional 1km GRIB2	1	60 MB	CLASS and NCEP
Total NVPS GVF	5	200MB	

3. SOFTWARE

This section first describe the system-level software elements invoked by the NDE production system. The detailed description of the entire NVPS design can be found in the appendix. Following the NVPS design, the document also provide a detailed description of sources codes and system files, which are organized in multi-level file structures. For each subdirectory, contents and their brief introductions are also available. This section ends with the source codes and their functionalities for each source code.

3.1. Software Description

NVPS consists of both GVF and VI subsystems. GVF subsystem requires inputs including daily granule TOC VIIRS red (I1), TOC VIIRS near-infrared (I2), and TOC VIIRS blue (M3) stored in NetCDF4 format and ancillary geolocation and water mask stored in h5 format. The outputs of GVF subsystem are daily rolling weekly GVF at global (0.036° and regional (0.009°) scales. VI subsystem requires inputs including daily granule TOA VIIRS red (I1), TOA VIIRS near-infrared (I2) reflectance bands stored in h5 format, TOC VIIRS red (I1), TOC VIIRS near-infrared (I2), and TOC VIIRS blue (M3), as well as quality assurance data (e.g., cloud mask, aerosol optical thickness (AOT)) stored in NetCDF4 format, and ancillary geolocation and water mask stored in h5 format.

The outputs of VI subsystem includes daily, weekly, and bi-weekly VI products with QA flags at global (0.036°) and regional (0.009°) scales. The final products of the two subsystems are available in NetCDF files. Tiff images of the final products are also produced for visualization.

NVPS GVF subsystem includes seven processing units: (1) the tile-granule mapping unit (TGM); (2) the surface reflectance gridded unit (GRD); (3) the surface reflectance compositing unit (SRC); (4) EVI calculating unit (CVI); (5) the EVI smoothing unit (SVI); (6) fine resolution NVPS calculating unit (GCL); (7) NVPS aggregation unit (GAG). The GRD unit calls TGM unit directly; each of other units is called by its own Bash script, which is invoked from an overall driver script.

TGM unit identifies which input granules fall on each of the 122 tiles used by the NVPS GVF intermediate processing. A granule may fall on several different tiles. The output is kept in temporary files, which are directly used by the subsequent unit GRD. No output filenames will be added to the PSF by TGM.

GRD unit grids all possible granule-level surface reflectances observations falling on a single tile into a tile. The filenames of the resultant surface reflectance tiles are written to the PSF and the files are kept for 7 days as input for the subsequent daily rolling weekly compositing.

SRC unit produces a daily rolling weekly surface reflectance composite from the gridded surface reflectance of the past 7 days by selecting the pixel that has the maximum view-angle-adjusted SAVI. The output is temporary and directly used in the following unit. No output filenames will be added into the PSF.

CVI unit calculates EVI from the surface reflectance composites. The filenames of output are written to PSF and the files are kept for 15 weeks, to be used in the smoothing of a time series of EVI.

SVI unit reduces the high-frequency noises in the EVI data by smoothing a time series of EVI of the past 15 weeks (including the current week). The smoothing occurs in two phases. The file names of output from phase-one are written to PSF and the files keep for 7 days to be used in the phase-two smoothing for the following 7 days. The output from phase-two will be directly used by the subsequent unit GCL and no filenames will be added to PSF.

GCL unit calculates NVPS GVF at the 0.003° resolution from the smoothed EVI data. The output will be directly used by GAG and no filenames will be added to PSF.

GAG unit aggregates the 0.003° NVPS GVF to 0.009° (regional product) and 0.036° (global product). For locations where there is NVPS GVF gap at 0.009° or 0.036°, NVPS GVF climatology is used. The final NVPS GVF products available in two NetCDF files. TIFF images of NVPS GVF are produced for visualization.

NVPS VI subsystem includes six processing units: (1) the tile-granule mapping unit, TGM, (2) the TOA and TOC reflectance gridding unit, GRD, (3) the TOA and TOC reflectance compositing unit, RCP , (4) TOA and TOC reflectance aggregating unit, GAG, (5) the TOA NDVI,TOC NDVI and TOC EVI calculating unit, VIC, (6) quality assurance assigning unit, QAA. The GRD unit calls TGM unit directly; each of other units is called by its own Bash script, which is invoked from an overall driver script. RCP unit need only be called to generate weekly and bi-weekly VI products at global (0.036°) and regional (0.009°) scales. Hence, in the process of generating daily VI products, VI subsystem will skip calling RCP unit.

VI retrieval algorithms require inputs including daily granule TOA VIIRS red (I1), TOA VIIRS near-infrared (I2) reflectance bands stored in h5 format, TOC VIIRS red (I1), TOC VIIRS near-infrared (I2), and TOC VIIRS blue (M3), as well as quality assurance data (e.g., cloud mask, aerosol optical thickness (AOT)) stored in NetCDF4 format, and ancillary geolocation and water mask stored in h5 format. The VI algorithm is carried out in following steps: 1) using geolocation information, grid and tile scheme to build tile-granule relationship; 2) mapping all available daily granule input data to the native VI geographic grid (0.003° plate carree projection) to produce a gridded daily TOA and TOC reflectance maps with quality assurance information; 3) if calculating daily VI, then skipping this step. Compositing the daily TOA and TOC reflectance maps of the last 7 and 16 days (including the current day) to produce a weekly and bi-weekly TOA and TOC reflectance maps using the MVA-SAVI compositing algorithm, which selects, at each VI grid point (pixel), the observation with maximum view-angle adjusted SAVI value in the 7-day or 16-day period. The 7-day and 16-day compositing is conducted with a daily rolling weekly and bi-weekly compositing scheme, which on each day uses data in the previous 6 and 15 days as inputs. Cloud mask information used in compositing and saved for the selected pixels in compositing; 4) aggregating daily, weekly and bi-weekly TOA and TOC reflectance maps from 0.003° to 0.009° and 0.036° resolution for output maps;

5) using NDVI and EVI schemes to calculate daily, weekly, and bi-weekly TOA NDVI, TOC NDVI and TOC EVI from the daily reflectance maps in step 2 and weekly and bi-weekly composited reflectance maps in step 3, respectively; 6) using spatial aggregation schemes to assess quality assurances of VI products mentioned in step 5.

TGM unit identifies which input granules fall on each of the 122 tiles used by the NVPS intermediate processing. A granule may fall on several different tiles. The output is kept in temporary files, which are directly used by the subsequent unit GRD. No output filenames will be added to the PSF by TGM.

GRD unit grids into each 0.003° -resolution tile with the TOA and TOC reflectance, cloud mask and aerosol optical thickness granules that fall on the tile. The filenames of the resultant TOA and TOC reflectance, cloud mask and aerosol optical thickness tiles are written to the PSF and the files are kept for 1 or 7 or 16 days as input for the subsequent daily aggregating or weekly or bi-weekly compositing.

RCP unit produces a daily rolling weekly or bi-weekly TOA and TOC reflectance, cloud mask and aerosol optical thickness composite from the gridded surface reflectance of the past 7 or 16 days by selecting the pixel that has the maximum view-angle-adjusted SAVI. The output is temporary and directly used the following unit. No output filenames to add to the PSF.

RAG unit aggregates the daily or weekly or bi-weekly 0.003° TOA and TOC reflectance, cloud mask and aerosol optical thickness to 0.009° (regional product) and 0.036°(global product).

VIC unit calculates daily or weekly or bi-weekly TOA NDVI, TOC NDVI, and TOC EVI from the TOA and TOC reflectance from outputs of RAG unit.

QAA unit assesses quality assurance of VI products from VIC unit using spatial aggregating scheme.

3.2. Directory Description

This section briefly describes the directory structure of the source code (C++ programs and bash scripts) and ancillary data that are delivered to NDE in the Delivered Algorithm Package (DAP). After the C++ source code for all software units are successfully compiled, a bash script is called to copy all the executables into a separate directory. This directory of executables will be copied into the production environment of NDE.

The directory in which subdirectories of source code reside is referred to as "*$BASE*". Source code is only kept on the test machine. Only executable code is allowed on the production machine.

$BASE/code/GranuleOnTile/ contains the g++ source code and Bash script for unit TGM, which identifies for each tile which granules fall on it.

$BASE/code/DailyGrid/ contains the g++ source code and Bash script for unit GRD, which grids daily surface reflectance.

$BASE/code/WeeklyComposite/ contains the g++ source code and Bash script for unit SRC, which generates a weekly surface reflectance composite from gridded daily surface reflectance.

$BASE/code/calcEVI/ contains the g++ source code and Bash script for unit CVI, which calculates EVI from daily rolling weekly surface reflectance composite.

$BASE/code/TSsmooth/ contains the g++ source code and Bash script for unit SVI, which smoothes the EVI time series in two phases.

$BASE/code/calcGVF/ contains the g++ source code and Bash script for unit GCL, which calculates fine-resolution NVPS from smoothed EVI.

$BASE/code/aggGVF/ contains the g++ source code and Bash script for unit GAG, which aggregates fine-resolution NVPS to create regional and global NVPS products.

$BASE/code/bin2tiff/ contains the g++ source code to make a TIFF image of each NVPS product by the GAG unit, for product visualization and production monitoring.

There are also two supporting subdirectories containing codes used by codes in other subdirectories. $BASE/code/bin/ contains C code and Bash script utilities such as date conversion. $BASE/code/common contains definition and implementation of many C++ classes that are shared by several software units.

$BASE/compile/ contains a script to set up the environmental variables and options for the compilation of the C++ source code, and a script to compile all the software units.

$BASE/run/ contains a driver script; Its beginning few lines are to be modified to suit the operational environment.

3.3. Source Code Description

As the NVPS source code consists of 81 C++ programs and Bash scripts, totaling approximately 10,000 lines of text, it is simply not possible to identify each program and describe its purpose in this section. Therefore, only the source code of each C++ main program is identified and its functionality described. All the C++ main programs have "_main.cpp" as the suffix of their filenames.

3.3.1. $BASE/code/GranuleOnTile/GranuleOnTile_main.cpp

This is the main program in the TGM unit. It takes a text file containing file names of pairs of VIIRS surface reflectance and geolocation granules to produce for each tile a text file containing a list of VIIRS granules that overlap with the tile. The resultant executable is called directly in script file and the output is used by the subsequent gridding unit.

This main program is supported by a few other programs in the same directory. Bash script *pairInput.sh* pairs, for a given date YYYYMMDD, each VIIRS surface reflectance granule with the corresponding geolocation granule, and the output is used as input to the main program. *GeoLoc.cpp* is the class implementation to handle (i.e., read) the geolocation information for a granule, and *AllGranule.cpp* is a container class implementation to handle the reflectance-geolocation granule pairs; They are links to files in the *common* directory and the links are made during the process of compiling.

3.3.2. $BASE/code/DailyGrid/DailyGrid_main.cpp

This is the main program in the GRD unit. For a given tile, it takes a text file containing a list of filenames of the VIIRS surface reflectance granules and the geolocation granule that overlap with the tile, and grids the surface reflectance at 0.003° resolution.

The griding code is called by script *run.daily.alltiles.sh* which runs gridding for all tiles in a loop; The script splits the list of tiles into several subsets and several instances of script *run.daily.subset.sh* runs on each subset in parallel to speed up the processing.

This main program is supported by a few other programs in the same directory.

WaterMask.cpp is the C++ class definition 0.003-deg water mask. *SurfRefl.cpp* handles the input VIIRS surface reflectance in granules. *SRTile.cpp* handles the reading/writing of gridded surface reflectance in tiles. These C++ files are links to files in *common* directory and are created during code compiling.

Script *move_granule_into_subdirectory.sh* moves VIIRS surface reflectance granules and geolocation granules from the working directory in the production environment into respective subdirectories, as the TGM unit requires certain subdirectory structure. A better place for this script would be $BASE/*code/GranuleOnTile,* but since *run.daily.alltiles.sh* call TGM and GRD in the same script, and TGM is actually transparent as a unit to in the external interface, it is kept in the $BASE/code/DailyGrid.

3.3.3. $BASE/code/WeeklyComposite/WeeklyComposite_main.cpp

This is the main program for the SRC unit. It takes the gridded surface reflectance of the most recent 7 days to make a weekly surface reflectance composite by selecting the best observation for each pixel.

Supporting programs include *SRTile.cpp (described in GRD), NVPSUtil.cpp* (utility functions) which are links to files in $BASE/code/common. Script *run.weekly.alltiles.sh* splits the set of 122 tiles into a few subsets, and calls *run.weekly.subset.sh* to run compositing on all subsets simultaneously. Script *run.weekly.alltiles.sh* is called by the driver script directly.

3.3.4. $BASE/code/calcEVI/calcEVI_main.cpp

This is the main program for a very simple unit CVI. It calculates enhanced vegetation index (EVI) from the daily rolling weekly surface reflectance composite. *EVI.cpp* is the implementation of the EVI C++ class. Script *run.calcEVI.sh* runs the calculation in a loop over all tiles, and it is called by the driver script.

3.3.5. $BASE/code/TSsmooth/TSsmooth_15wk.cpp, and

$BASE/ code/TSsmooth/TSsmooth_movave.cpp

The EVI smoothing unit SVI is a relatively complex unit and the smoothing is accomplished in two phases. The first phase is by *TSsmooth_15wk.cpp*, which takes the EVI times series consisting of EVI values of the most recent 15 (non-overlapping) weeks to produce a smoothed version of the EVI time series for each 0.003^o pixel; Save the smoothed data for the most recent day as phase-one smoothed result. The second phase is by *TSsmooth_movave.cpp*, which averages the phase-one EVI results of the most recent 7 days as the final smoothed EVI for the current day.

3.3.6. $BASE/code/calcNVPS/calcNVPS_main.cpp

This is the main program for a very simple unit, GCL. It calculates NVPS at 0.003° resolution from the smoothed EVI.

Script *run.calcNVPS.sh* does the calculation for all tiles.

3.3.7. $BASE/code/ bin2tiff/ NVPStiff_main.cpp

This program presents the NVPS values in a GeoTIFF file. The color lookup table is VI_global.dsr in the same directory. It is called by the GAG unit (next).

3.3.8. $BASE/code/aggNVPS/aggNVPS_main.cpp

This main program in the GAG unit aggregates the 0.003° NVPS in HDF5 to the 0.009° regional NVPS product and 0.036° global NVPS product, both in NetCDF. It also computes NVPS statistics for selected areas, calls NVPSTiff_main to make a GeoTIFF color image of NVPS, for visualization and production monitoring.

4. NORMAL OPERATIONS

4.1. System Control Files

Current NVPS VI & GVF are independently implemented in NDE. Each of NVPS VI & GVF systems communicates with the external environment via three system control files – a process control file (PCF) to specify the input and set the parameters to run the system, a process status file (PSF) to record the names of the generated files to be saved, and a log file (LOG) to track the progress of the processing, and errors if there are any. All the software units access the same PCF, PSF and LOG files; There are no separate control files for each unit.

Before the NDE system invokes the NVPS driver script, it creates a working directory and a PCF specific to the given run. The PCF is placed in the working directory, and so are the daily input granule data, and the intermediate files produced by previous runs if there is any. Then an instance of the driver script is run by NDE to generate NVPS products. As the driver script runs, names of intermediate and final files to be kept are written to PSF. The progress of the processing is echoed into the LOG file. When the driver script finishes successfully, it returns a value zero as exit status. Any non-zero exit status signals an error which causes premature termination of the process, and the log file reports the source of the error.

4.1.1. Process Control File (PCF)

The PCF for VI contains the names to geolocation granule GITCO filenames, the daily input top of atmosphere reflectance granule SVI01&2 file names, the daily input surface reflectance granule SR file names, the daily input aerosol optical depth granule JRR-AOD file names, and the daily input cloud mask granule JRR-CloudMask filenames. The PCF for GVF product contains the names to geolocation granule GITCO filenames, the daily input surface reflectance granule SR filenames, and intermediate files generated from previous runs. A less dynamic part of PCF contains the filenames of the water masks, NVPS climatology files, and other ancillary data. The number of intermediate files is huge, as in internal processing, the NVPS system divide the study region into hundreds of tiles and there are different types of files associated with each tile. The PCF is best explained with an example, but since there are approximately 1000 input granules each day and there are 122 NVPS tiles, a complete PCF file will have about 5000 lines. The following example shows a small sample of input granules and intermediate files only for one of the 122 tiles.

```
#
# name: /opt/data/nde/NDE_DEV3/pgs/working/2/viirs_vi.pcf
#
# working directory: /opt/data/nde/NDE_DEV3/pgs/working/2
# ProdRuleName: VI v1.2 Temporal PR
#
```

working_directory=/opt/data/nde/NDE_DEV3/pgs/working/2

nde_mode=NDE_DEV3

job_coverage_start=201508190000000

job_coverage_end=201508200000000

CODE_PAR_DIR=/opt/apps/nde/NDE_DEV3/algorithms/VI/v1.2/ops

WATERMASK_DIR=/opt/apps/nde/NDE_DEV3/algorithms/VI/v1.2/watermask

NVPSCLIMAT_GLOBAL=/opt/apps/nde/NDE_DEV3/algorithms/VI/v1.2/gvfclimat/VIIRS_global_GVF_climatology.h5

NVPSCLIMAT_REGIONAL=/opt/apps/nde/NDE_DEV3/algorithms/VI/v1.2/gvfclimat/VIIRS_regional_GVF_climatology.h5

NP=5

VER_REV=v1r2

CHOICE_GEN_GEOTIFF=1

INPUT_FILE_TYPES=GITCO SVI01 SVI02 SR JRR-AOD JRR-CloudMask

Input terrain corrected geolocation granules. A small sample.

INPUT=GITCO_npp_d20150819_t0009564_e0011206_b19729_c20170401043243730257_noaa_ops.h5

INPUT=GITCO_npp_d20150819_t0011218_e0012460_b19729_c20170401043243730257_noaa_ops.h5

INPUT=GITCO_npp_d20150819_t0012472_e0014114_b19729_c20170401043257904759_noaa_ops.h5

Input TOA reflectance (SVI01). A small sample.

INPUT=SVI01_npp_d20150819_t0011218_e0012460_b19729_c20170331173324908536_noaa_ops.h5

INPUT=SVI01_npp_d20150819_t0012472_e0014114_b19729_c20170331173325693471_noaa_ops.h5

INPUT=SVI01_npp_d20150819_t0014126_e0015368_b19729_c20170331173325693471_noaa_ops.h5

Input TOA reflectance (SVI02). A small sample.

INPUT=SVI02_npp_d20150819_t0011218_e0012460_b19729_c20170331000225131003_noaa_ops.h5

INPUT=SVI02_npp_d20150819_t0012472_e0014114_b19729_c20170331000235452592_noaa_ops.h5

INPUT=SVI02_npp_d20150819_t0014126_e0015368_b19729_c20170331000235452592_noaa_ops.h5

Input surface reflectance granules. A small sample.

INPUT=SR_v1-0-8_npp_s201508190009564_e201508190011206_c201705121848420.nc

INPUT=SR_v1-0-8_npp_s201508190011218_e201508190012460_c201705121849090.nc

INPUT=SR_v1-0-8_npp_s201508190012472_e201508190014114_c201705121849540.nc

Input JRR-AOD granules. A small sample.

INPUT=JRR-AOD_v1r1_npp_s201508190009564_e201508190011206_c201704172121100.nc

INPUT=JRR-AOD_v1r1_npp_s201508190011218_e201508190012460_c201704172128380.nc

INPUT=JRR-AOD_v1r1_npp_s201508190012472_e201508190014114_c201704172136030.nc

Input JRR-CloudMask granules. A small sample.

INPUT=JRR-CloudMask_v1r1_npp_s201508190009564_e201508190011206_c201704172121000.nc

INPUT=JRR-CloudMask_v1r1_npp_s201508190011218_e201508190012460_c201704172128280.nc

INPUT=JRR-CloudMask_v1r1_npp_s201508190012472_e201508190014114_c201704172135540.nc

production_environment=DE

production_site=NSOF

#END-of-PCF

```
#
# name: /opt/data/nde/NDE_DEV3/pgs/working/1/viirs_gvf.pcf
#
# working directory: /opt/data/nde/NDE_DEV3/pgs/working/1
# ProdRuleName: GVF Temporal Rule v2.1
#
working_directory=/opt/data/nde/NDE_DEV3/pgs/working/1
nde_mode=NDE_DEV3
job_coverage_start=201508190000000
job_coverage_end=201508200000000
WATERMASK_DIR=/opt/apps/nde/NDE_DEV3/algorithms/GVF/v2.1/watermask
GVFCLIMAT_GLOBAL=/opt/apps/nde/NDE_DEV3/algorithms/GVF/v2.1/gvfclimat/VIIRS_global_GVF_climatology.h5
GVFCLIMAT_REGIONAL=/opt/apps/nde/NDE_DEV3/algorithms/GVF/v2.1/gvfclimat/VIIRS_regional_GVF_climatology.h5
CODE_PAR_DIR=/opt/apps/nde/NDE_DEV3/algorithms/GVF/v2.1/GVFEXEDIR
NP=2
VER_REV=v2r1
CHOICE_GEN_GEOTIFF=1
INPUT_FILE_TYPES=GITCO SR

# Input terrain corrected geolocation granules. A small sample.
```

INPUT=GITCO_npp_d20150819_t0009564_e0011206_b19729_c201704010432
43730257_noaa_ops.h5

INPUT=GITCO_npp_d20150819_t0011218_e0012460_b19729_c201704010432
43730257_noaa_ops.h5

INPUT=GITCO_npp_d20150819_t0012472_e0014114_b19729_c201704010432
57904759_noaa_ops.h5

Input surface reflectance granules. A small sample.

INPUT=SR_v1-0-
8_npp_s201508190009564_e201508190011206_c201705121848420.nc

INPUT=SR_v1-0-
8_npp_s201508190011218_e201508190012460_c201705121849090.nc

INPUT=SR_v1-0-
8_npp_s201508190012472_e201508190014114_c201705121849540.nc

production_environment=DE

production_site=NSOF

#END-of-PCF

4.1.2. Process Status File (PSF)

The PSF for VI contains the names of the gridded surface reflectance files of the current day, the TOC-EVI,TOC-NDVI, TOA-NDVI files for daily, the current daily rolling weekly composite, and the current daily rolling biweekly composite. The example files for tile h06v05 are provided below.

#

viirs_vi.psf

#

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_sr/20150819/VI-SR_s20150819_e20150819_h06v05_c201803192047160.h5

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-DLY-GLB_v1r2_npp_s20150819_e20150819_c201803192339160.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-DLY-REG_v1r2_npp_s20150819_e20150819_c201803200030110.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-DLY-GLB_v1r2_npp_s20150819_e20150819_c201803192339160_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-DLY-REG_v1r2_npp_s20150819_e20150819_c201803200030110_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-EVI-DLY-GLB_v1r2_npp_s20150819_e20150819_c201803192339160.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-EVI-DLY-REG_v1r2_npp_s20150819_e20150819_c201803200030110.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-TOA-NDVI-DLY-GLB_v1r2_npp_s20150819_e20150819_c201803192339160.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-TOA-NDVI-DLY-REG_v1r2_npp_s20150819_e20150819_c201803200030110.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-TOC-NDVI-DLY-GLB_v1r2_npp_s20150819_e20150819_c201803192339160.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/daily_aasr/20150819/VI-TOC-NDVI-DLY-REG_v1r2_npp_s20150819_e20150819_c201803200030110.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-WKL-GLB_v1r2_npp_s20150813_e20150819_c201803200234040.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-WKL-REG_v1r2_npp_s20150813_e20150819_c201803200322120.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-WKL-GLB_v1r2_npp_s20150813_e20150819_c201803200234040_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-WKL-REG_v1r2_npp_s20150813_e20150819_c201803200322120_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-EVI-WKL-GLB_v1r2_npp_s20150813_e20150819_c201803200234040.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-EVI-WKL-REG_v1r2_npp_s20150813_e20150819_c201803200322120.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-TOA-NDVI-WKL-GLB_v1r2_npp_s20150813_e20150819_c201803200234040.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-TOA-NDVI-WKL-REG_v1r2_npp_s20150813_e20150819_c201803200322120.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-TOC-NDVI-WKL-GLB_v1r2_npp_s20150813_e20150819_c201803200234040.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/weekly_aasr/20150813-20150819/VI-TOC-NDVI-WKL-REG_v1r2_npp_s20150813_e20150819_c201803200322120.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-BWKL-GLB_v1r2_npp_s20150804_e20150819_c201803200406440.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-BWKL-REG_v1r2_npp_s20150804_e20150819_c201803200454380.nc

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-BWKL-GLB_v1r2_npp_s20150804_e20150819_c201803200406440_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-BWKL-REG_v1r2_npp_s20150804_e20150819_c201803200454380_stat.txt

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-EVI-BWKL-GLB_v1r2_npp_s20150804_e20150819_c201803200406440.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-EVI-BWKL-REG_v1r2_npp_s20150804_e20150819_c201803200454380.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-TOA-NDVI-BWKL-GLB_v1r2_npp_s20150804_e20150819_c201803200406440.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-TOA-NDVI-BWKL-REG_v1r2_npp_s20150804_e20150819_c201803200454380.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-TOC-NDVI-BWKL-GLB_v1r2_npp_s20150804_e20150819_c201803200406440.tif

/opt/data/nde/NDE_DEV3/pgs/working/2/biweekly_aasr/20150804-20150819/VI-TOC-NDVI-BWKL-REG_v1r2_npp_s20150804_e20150819_c201803200454380.tif

The PSF for GVF contains the names of the gridded surface reflectance files of the current day, the EVI files for the current daily rolling week directly calculated from the current daily rolling weekly composite (i.e., before-smoothing EVI), and EVI files from phase-one time series smoothing. The example files for tile h06v05 are given below.

```
#
# viirs_gvf.psf
#
daily_sr/20150819/GVF-
```

SR_s20150819_e20150819_h06v05_c201803201830100.h5

weekly_bsevi/20150813-20150819/GVF-EVI_s20150813_e20150819_h06v05_c201803202022570.h5

weekly_asevi/15week/20150813-20150819/GVF-ASEVI-P1_s20150813_e20150819_h06v05_c201803202049240.h5

weekly_aagvf/20150813-20150819/GVF-WKL-GLB_v2r1_npp_s20150813_e20150819_c201803202143490.nc

weekly_aagvf/20150813-20150819/GVF-WKL-REG_v2r1_npp_s20150813_e20150819_c201803202147000.nc

weekly_aagvf/20150813-20150819/GVF-WKL-GLB_v2r1_npp_s20150813_e20150819_c201803202143490.tif

weekly_aagvf/20150813-20150819/GVF-WKL-REG_v2r1_npp_s20150813_e20150819_c201803202147000.tif

weekly_aagvf/20150813-20150819/GVF-WKL-GLB_v2r1_npp_s20150813_e20150819_c201803202143490_stat.txt

weekly_aagvf/20150813-20150819/GVF-WKL-REG_v2r1_npp_s20150813_e20150819_c201803202147000_stat.txt

#END-of-PSF

4.1.3. Process Log File

Upon entering a new processing unit, the NVPS system records that transition information into the log file. On rare occasions of error, the log file also records a detailed description of the errors.

4.2. Installation

The most import step in NVPS system installation is to update the configuration file regarding the library paths and compilation options.

4.2.1. Setup Requirements

This section lists all the items that are delivered as part of the NVPS DAP.

All source code is located in the *code* subdirectory (*$BASE/code/*), which is described in section 3.2*; Makefiles are in the same subdirectories that the main programs are in.*

The scripts to compile the code in $BASE/compile/. This directory also has a script which copies the executables in a directory, to separate from the source code.

Example PCF and driver script in $BASE/run/

Ancillary data are delivered separately from the code due to its large data volume. In the operational environment, the ancillary water masks in individual tiles are currently saved at $BASE/water/, and the two NVPS climatology files at global and regional scales are saved in $BASE/climatology/.

4.2.2. 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 a 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 files 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 production 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 4.2.1 are installed, NVPS System Software can be installed the following way.

(1) Copy DAP and build instructions to Linux build machine.

(2) Unpacking the system package in the *$BASE* directory

tar –xvzf VIIRS_NVPS_v1r0_20171203.tar.gz

This will create 4 subdirectories in the $BASE directory.

code/ - location of all source code (e.g.C/C++ code, scripts).

compile/ - location of scripts to compile the code and to create a directory of executables. Script export_linux_path.sh sets up the library paths and compiling options. export_linux_path.sh,   compile.sh compiles all the code, and copy_exe.sh copies all the executables from the source code directories to a designated directory.

run/ sample driver's script and PCF.

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

3) Update the library paths and compiling options if necessary.

For example, shortly after the delivery of the DAP, the NVPS-required libraries were rebuilt and therefore $BASE/compile/export_linux_path.sh was updated accordingly:

```
### TIFF
export TIFFINC=/opt/apps/ots/tiff/tiff-4.0.3/include
export TIFFLIB=/opt/apps/ots/tiff/tiff-4.0.3/lib

### GeoTIFF
export GEOTIFFINC=/opt/apps/ots/libgeotiff/libgeotiff-1.4.0/include
export GEOTIFFLIB=/opt/apps/ots/libgeotiff/libgeotiff-1.4.0/lib

export TIFFLINK="-static -ltiff -lgeotiff -lz -lm"
```

Additionally, the makefile $BASE/code/bin2tiff/Makefile needs updating by adding $(ZLIB) library to paths:

```
$(TARGET): $(OBJ) $(CC) -o $(TARGET) $(OBJ) -L$(ZLIB) -L$(TIFFLIB) \
```

-L$(GEOTIFFLIB) $(TIFFLINK) $(OPT)

4) compile

cd $BASE/compile and run

sh ./compile.sh

Then

sh ./copy_exe.sh

to copy the executables to a directory named NVPSEXEDIR which is parallel to source code directory "code."

5) Deploy application by copying the following folders into the NVPS version algorithm folder on each Linux processor

```
cp -rp run                algorithms/NVPS/v1.0/
cp -rp NVPSEXEDIR  algorithms/NVPS/v1.0/
cp -rp NVPSclimat  algorithms/NVPS/v1.0/
cp -rp watermask  algorithms/NVPS/v1.0/
```

4.3. Build

4.3.1. Compilation Procedures

All NOAA Unique Product (NUP) algorithms are compiled on the AIX operating system statically linked to the standard set of XL compilers. In the near future, algorithms will be migrated to the Dell Blade servers running Redhat Enterprise Linux (RHEL). In the Linux environment, algorithms can statically link to the standard set of GNU Compiler Collection (GCC) compilers or the Intel compilers for Linux.

NUP science algorithms are compiled on dedicated build machines under Configuration Management (CM) control. There is an AIX Power 6 Series build server and a Dell Blade build server running RHEL. When compiling algorithms on the Power 6 build machine, care must be taken to set XL compiler option for “qarch” to “pwr6” which enables the statically built binaries to run on both Power 6 and Power 7 Series. The NDE Test and Dev strings have Power 6 Series machines rather than Power 7 as in Production.

4.3.2. 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.

4.4. Operations Procedures

4.4.1. Normal Operations

4.4.2. Data Preparation

The primary source of input data for NDE is the Joint Polar Satellite System (JPSS)

NDE. The NDE 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, NDE 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 NDE xDR file (after NDE generates its 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 ingesting, NDE reads particular metadata fields from the xDR files. 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 (ESPD 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 are 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.

4.5. Distribution

4.5.1. 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 8 distribution servers, 4 for push customers and 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).

4.5.2. 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 while non real-time users may order the data from the CLASS.

4.5.3. 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. 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.

4.5.4. External Product Tools

No external product tools are supplied. The NVPS 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.

5. MONITORING AND MAINTENANCE

5.1. 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 (GUI) available on Sandia. This interface works as intranet and requires a “Process Status” text file from the NVPS production server. The NVPS operational server will push the process status file to the NVPS QA server for products monitoring and data quality assurance. .

5.2. Product Monitoring

NVPS 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 NVPS data Quality Monitoring. Once the NVPS products are successfully generated, NVPS statistics at a few selected sites are written to a text file and a TIFF image of color-code NVPS values is generated for both the global and regional products. Abrupt change in the NVPS text output indicates a potential problem. And gap in the TIFF image also signals partial failure of the production system.

5.3. Maintenance

5.3.1. 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 Users 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, specifically servers (green color indicates system running normally while red color code suggests system failure). 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 screenshot). 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.

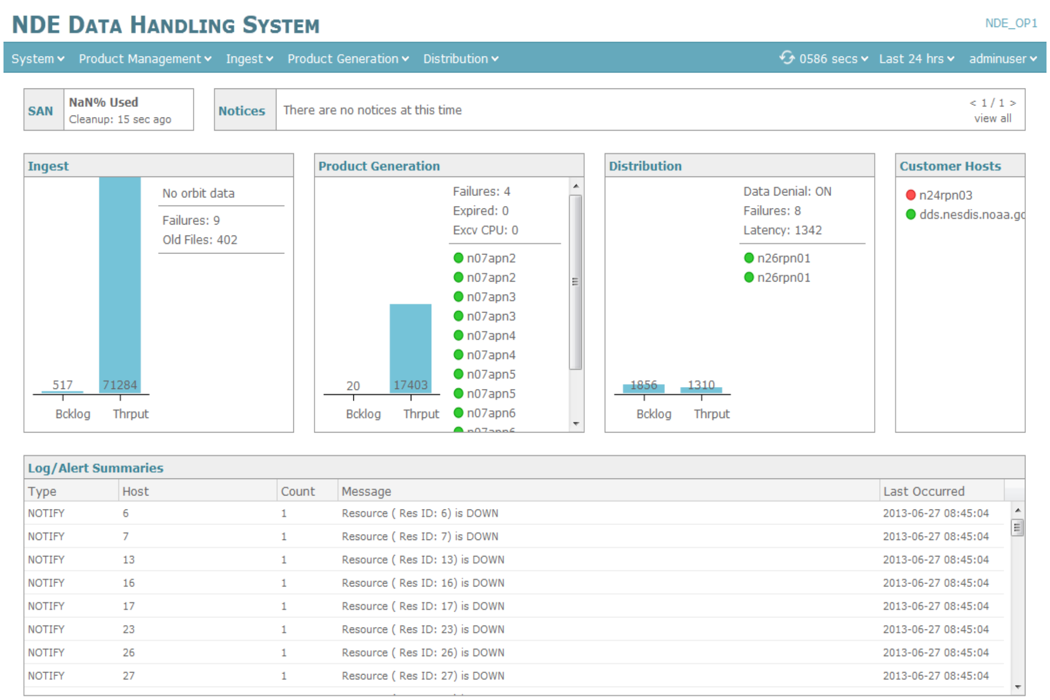


Figure 14. NDE DHS Dashboard GUI Page.

5.3.2. Ingest Monitoring

Monitoring of ingest can be accomplished using the Ingest Backlog and Throughput page (below). This page shows the backlog in the lading 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.

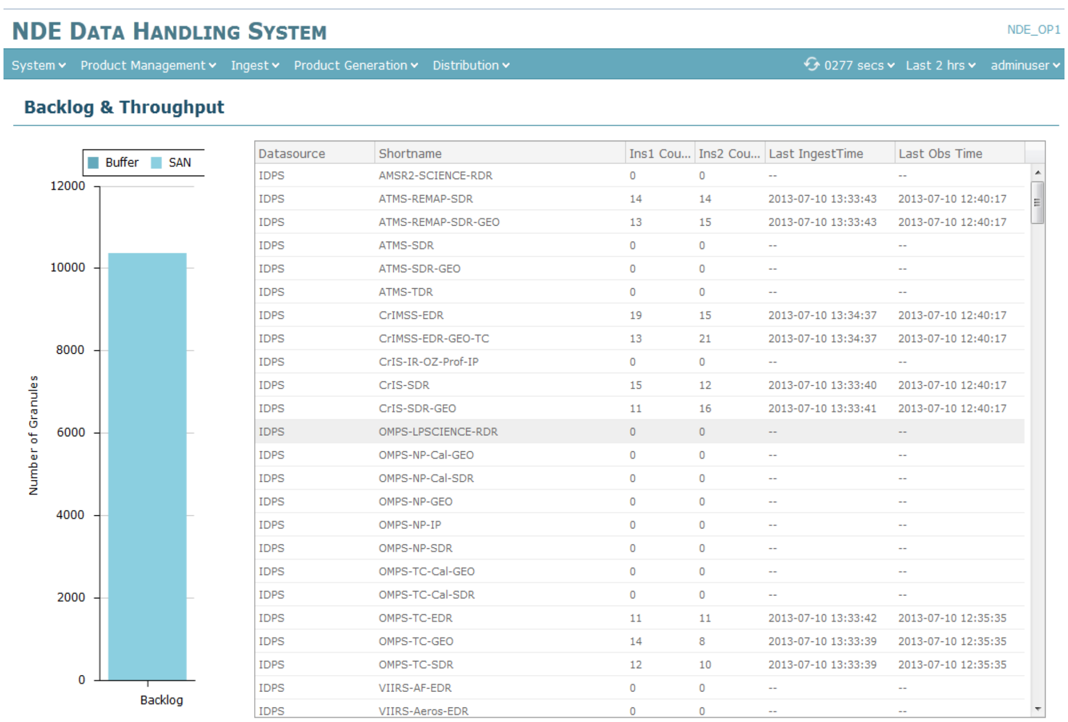


Figure 15. NDE DHS Backlog and Throughput GUI Page.

5.3.3. 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, and 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 have 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, 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 have 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. If the algorithm returns with a non-zero code or simply errors out (e.g. core dump), 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, 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 located.

Table 11. 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 have not been ingested within a predefined period of time (as defined in the	Production Job Spec is done – no transition from this state.

	production rule).	

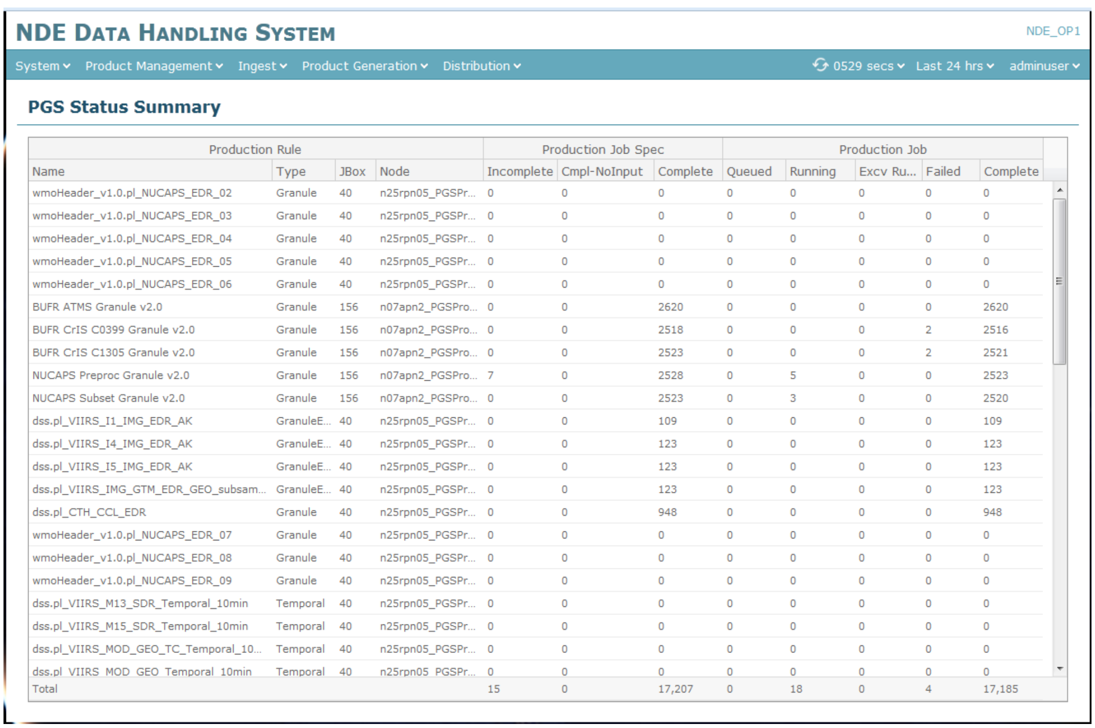
Table 12. Production Job States.

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.
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	Processing node

	a zero code and written output file names to the Process Status File (PSF). NDE DHS reads the PSF file names to be ingested.	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.

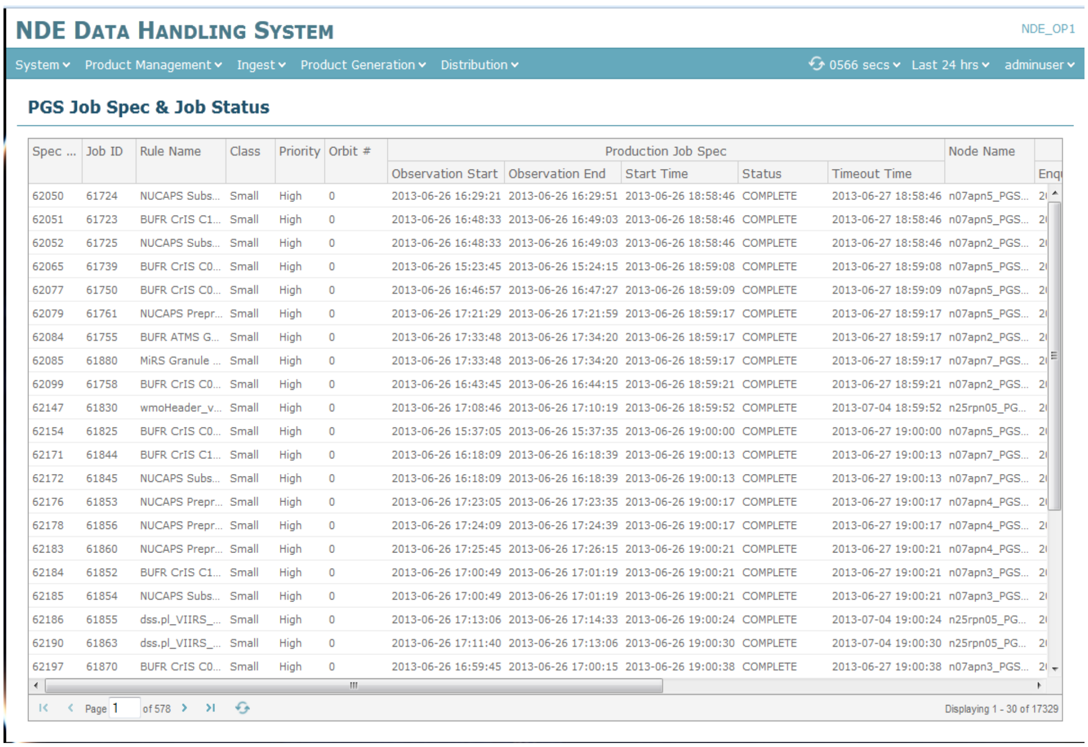
The Product Generation drop down menu has a link to the PGS Status Summary, which represents a tally of all the current states a particular production rule is in. The figure below shows the status page. All states are clickable and will bring up the specific production job specs or production jobs for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



Production Rule				Production Job Spec			Production Job				
Name	Type	JBox	Node	Incomplete	Cmpl-NoInput	Complete	Queued	Running	Excv Ru...	Failed	Complete
wmoHeader_v1.0.pl_NUCAPS_EDR_02	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_03	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_04	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_05	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_06	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
BUFR ATMS Granule v2.0	Granule	156	n07apn2_PGSPro...	0	0	2620	0	0	0	0	2620
BUFR CrIS C0399 Granule v2.0	Granule	156	n07apn2_PGSPro...	0	0	2518	0	0	0	2	2516
BUFR CrIS C1305 Granule v2.0	Granule	156	n07apn2_PGSPro...	0	0	2523	0	0	0	2	2521
NUCAPS Preproc Granule v2.0	Granule	156	n07apn2_PGSPro...	7	0	2528	0	5	0	0	2523
NUCAPS Subset Granule v2.0	Granule	156	n07apn2_PGSPro...	0	0	2523	0	3	0	0	2520
dss.pl_VIIRS_I1_IMG_EDR_AK	GranuleE...	40	n25rpn05_PGSPr...	0	0	109	0	0	0	0	109
dss.pl_VIIRS_I4_IMG_EDR_AK	GranuleE...	40	n25rpn05_PGSPr...	0	0	123	0	0	0	0	123
dss.pl_VIIRS_I5_IMG_EDR_AK	GranuleE...	40	n25rpn05_PGSPr...	0	0	123	0	0	0	0	123
dss.pl_VIIRS_IMG_GTM_EDR_GEO_subsam...	GranuleE...	40	n25rpn05_PGSPr...	0	0	123	0	0	0	0	123
dss.pl_CTH_CCL_EDR	Granule	40	n25rpn05_PGSPr...	0	0	948	0	0	0	0	948
wmoHeader_v1.0.pl_NUCAPS_EDR_07	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_08	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
wmoHeader_v1.0.pl_NUCAPS_EDR_09	Granule	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
dss.pl_VIIRS_M13_SDR_Temporal_10min	Temporal	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
dss.pl_VIIRS_M15_SDR_Temporal_10min	Temporal	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
dss.pl_VIIRS_MOD_GEO_TC_Temporal_10...	Temporal	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
dss.pl_VIIRS_MOD_GEO_Temporal_10min	Temporal	40	n25rpn05_PGSPr...	0	0	0	0	0	0	0	0
Total				15	0	17,207	0	18	0	4	17,185

Figure 16. 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.



Spec ...	Job ID	Rule Name	Class	Priority	Orbit #	Production Job Spec					Node Name
						Observation Start	Observation End	Start Time	Status	Timeout Time	
62050	61724	NUCAPS Subs...	Small	High	0	2013-06-26 16:29:21	2013-06-26 16:29:51	2013-06-26 18:58:46	COMPLETE	2013-06-27 18:58:46	n07apn5_PGS...
62051	61723	BUFR CrIS C1...	Small	High	0	2013-06-26 16:48:33	2013-06-26 16:49:03	2013-06-26 18:58:46	COMPLETE	2013-06-27 18:58:46	n07apn5_PGS...
62052	61725	NUCAPS Subs...	Small	High	0	2013-06-26 16:48:33	2013-06-26 16:49:03	2013-06-26 18:58:46	COMPLETE	2013-06-27 18:58:46	n07apn2_PGS...
62065	61739	BUFR CrIS C0...	Small	High	0	2013-06-26 15:23:45	2013-06-26 15:24:15	2013-06-26 18:59:08	COMPLETE	2013-06-27 18:59:08	n07apn5_PGS...
62077	61750	BUFR CrIS C0...	Small	High	0	2013-06-26 16:46:57	2013-06-26 16:47:27	2013-06-26 18:59:09	COMPLETE	2013-06-27 18:59:09	n07apn5_PGS...
62079	61761	NUCAPS Prepr...	Small	High	0	2013-06-26 17:21:29	2013-06-26 17:21:59	2013-06-26 18:59:17	COMPLETE	2013-06-27 18:59:17	n07apn5_PGS...
62084	61755	BUFR ATMS G...	Small	High	0	2013-06-26 17:33:48	2013-06-26 17:34:20	2013-06-26 18:59:17	COMPLETE	2013-06-27 18:59:17	n07apn2_PGS...
62085	61880	MiRS Granule ...	Small	High	0	2013-06-26 17:33:48	2013-06-26 17:34:20	2013-06-26 18:59:17	COMPLETE	2013-06-27 18:59:17	n07apn7_PGS...
62099	61758	BUFR CrIS C0...	Small	High	0	2013-06-26 16:43:45	2013-06-26 16:44:15	2013-06-26 18:59:21	COMPLETE	2013-06-27 18:59:21	n07apn2_PGS...
62147	61830	wmoHeader_v...	Small	High	0	2013-06-26 17:08:46	2013-06-26 17:10:19	2013-06-26 18:59:52	COMPLETE	2013-07-04 18:59:52	n25rpn05_PG...
62154	61825	BUFR CrIS C0...	Small	High	0	2013-06-26 15:37:05	2013-06-26 15:37:35	2013-06-26 19:00:00	COMPLETE	2013-06-27 19:00:00	n07apn5_PGS...
62171	61844	BUFR CrIS C1...	Small	High	0	2013-06-26 16:18:09	2013-06-26 16:18:39	2013-06-26 19:00:13	COMPLETE	2013-06-27 19:00:13	n07apn7_PGS...
62172	61845	NUCAPS Subs...	Small	High	0	2013-06-26 16:18:09	2013-06-26 16:18:39	2013-06-26 19:00:13	COMPLETE	2013-06-27 19:00:13	n07apn7_PGS...
62176	61853	NUCAPS Prepr...	Small	High	0	2013-06-26 17:23:05	2013-06-26 17:23:35	2013-06-26 19:00:17	COMPLETE	2013-06-27 19:00:17	n07apn4_PGS...
62178	61856	NUCAPS Prepr...	Small	High	0	2013-06-26 17:24:09	2013-06-26 17:24:39	2013-06-26 19:00:17	COMPLETE	2013-06-27 19:00:17	n07apn4_PGS...
62183	61860	NUCAPS Prepr...	Small	High	0	2013-06-26 17:25:45	2013-06-26 17:26:15	2013-06-26 19:00:21	COMPLETE	2013-06-27 19:00:21	n07apn4_PGS...
62184	61852	BUFR CrIS C1...	Small	High	0	2013-06-26 17:00:49	2013-06-26 17:01:19	2013-06-26 19:00:21	COMPLETE	2013-06-27 19:00:21	n07apn3_PGS...
62185	61854	NUCAPS Subs...	Small	High	0	2013-06-26 17:00:49	2013-06-26 17:01:19	2013-06-26 19:00:21	COMPLETE	2013-06-27 19:00:21	n07apn3_PGS...
62186	61855	dss.pl_VIIRS_...	Small	High	0	2013-06-26 17:13:06	2013-06-26 17:14:33	2013-06-26 19:00:24	COMPLETE	2013-07-04 19:00:24	n25rpn05_PG...
62190	61863	dss.pl_VIIRS_...	Small	High	0	2013-06-26 17:11:40	2013-06-26 17:13:06	2013-06-26 19:00:30	COMPLETE	2013-07-04 19:00:30	n25rpn05_PG...
62197	61870	BUFR CrIS C0...	Small	High	0	2013-06-26 16:59:45	2013-06-26 17:00:15	2013-06-26 19:00:38	COMPLETE	2013-06-27 19:00:38	n07apn3_PGS...

Figure 17. NDE DHS PGS Job Spec & Job Status GUI Page

The detailed window shows all input data for a particular job (Figure 18). 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.

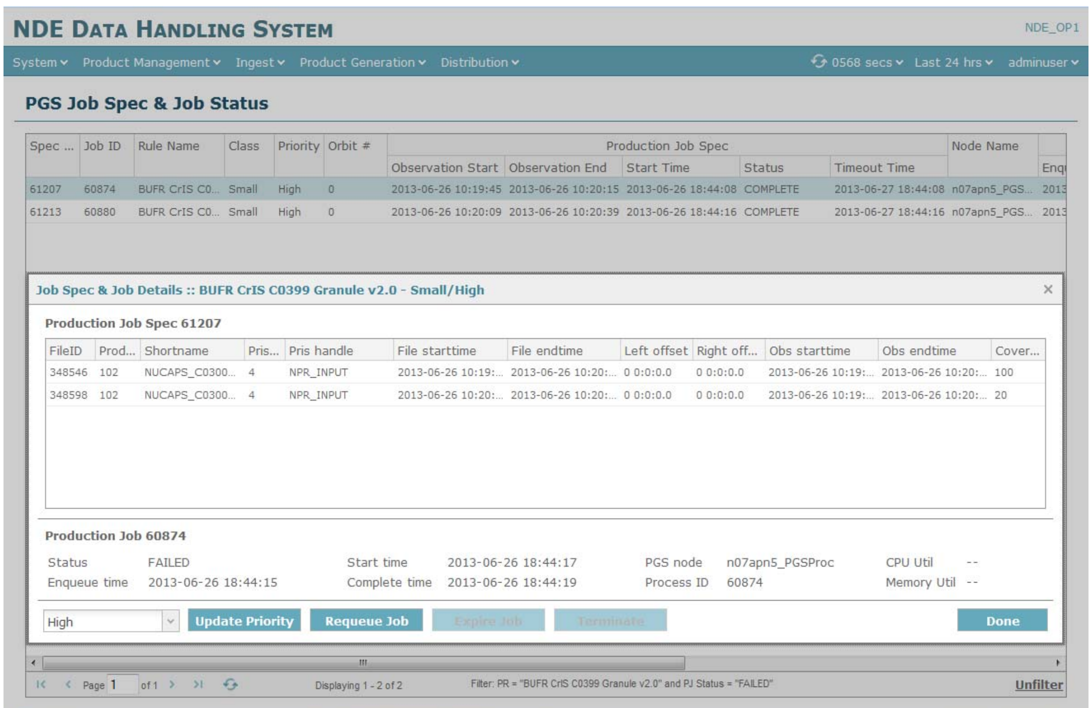


Figure 18. NDE DHS PGS Job Spec & Job Status Details GUI Page

5.3.4. 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 13. Distribution Prepare Request States.

States	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 creates links to a subscribed product file in the dist directory on the SAN. 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 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
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 14. Push Distribution Job States.

State	Initialization	Transition
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 pushed to its destination.	No transition from this state without user intervention.

5.3.5. 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 the outputs.

5.3.6. 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.

5.3.7. Special Procedures

Installation of new DHS Build is required to implement new features or to repair current features of the DHS.

5.3.8. 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.

5.4. 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 NVPS production system (PE-1), including the data required for processing. Alternative backup for NVPS operational system can be the NVPS test system that is operating at the NDE. For a mission critical, data 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.

6. TROUBLESHOOTING

6.1. Problem Diagnosis and Recovery

6.1.1. Error Correction

All NDE Data Handling System (DHS) applications are Java-based and built using the Enterprise Service Bus (ESB) framework provided by JBoss. NDE DHS consists of five ESB applications, which perform the core tasks for ingestion, 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 15. DHS Error Messages

Message	Subsystem Application	Reason	What to do?
Execute Checks : Exception thrown	Ingest	An error occurred during the validation of a product file being	Check to make sure h5dump (or other relevant MD extractors) is installed on the

		ingested. Error could be during metadata extraction or when checksums don't match.	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	Check to make sure LoopName attribute in properly defined in the ESB configuration file (jboss-esb.xml) of the

		jboss-esb.xml)	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	Ingest	Ingest app unable to retrieve rows from IngestRequestLog table for a given	Make sure the DB is functioning normally. If it is, a TAL must determine the cause of the inconsistencies

objects		IR_ID. May be a DB error or the IRB and IRL tables must be out of sync.	between IRL and IRB table.
validateFilenameP attern : Exception occured during validation of the product file	Ingest		
validateProduct : Product File Unknown	Ingest	The product file being ingested is of unknown product. Their 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	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.

		ingested	
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 -	PGS	App unable to execute stored	Same as above.

Exception occurred	Factory	proc (SP_GET_FILE_ ACCUMULATION) to retrieve the File accumulation thresholds for a ProductionJobSp ec.	
createJobSpecInp ut : Exception detected	PGS Factory	App unable to create (insert into DB) a Job Spec Input for a ProductionJobSp ec.	Make sure DB is OK. If it is, then follow the Exception 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.
generateChecksu m : 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	Make sure the DB is functioning normally. If it is, follow the Exception's stack trace to determine the root cause of the Exception.

		executed.	
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 :	PGS	The app is unable to copy the files	Make sure the files in question exist in

Exception occurred	Processor	(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.	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	PGS	The app is unable	Make sure the

runProdJob Action	Processor	to execute an algorithm for whatever reason.	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	Make sure the DB is functioning normally. If it is then follow the exception's stack trace to determine the root cause.

		times as it changes from one state to another.	
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	Make sure stored proc is registered in DB and DB is functioning normally.

		from DB.	
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	DIST ,PGS Processor	The app is unable to update the Job Status to Fail in	Make sure database is functioning normally. If it is, then follow the

Status to Fail		DB.	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	Same as above.

		unexpectedly.	
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.

6.1.2. Problem Diagnosis and Recovery Procedures

The NVPS system is designed to be robust. If there is an error in the input files, the NVPS system records the error into the log file, and skips the erroneous input to continue with the next file. At the final stage of NVPS production, any data gap is filled with climatology data. Therefore the NVPS system is expected to run smoothly without external intervention.

If there are serious problems (e.g., out of memory) that prevents the operational run, a detailed description of the errors will be written into the log file (e.g., line number in the source code, the function name, and the source code filename), and then the code terminates with non-zero exit status. In this case, maintenance personnel intervention is required.

6.1.3. Program Recovery Procedures

Operational algorithms are assigned across multiple nodes. If a node fails, an alternative node will continue the processing without the need of operator intervention. If the entire system fails, job specifications can be requeued and restarted. If necessary the database will failover to the backup database.

6.2. Application Shutdown and Restart

6.2.1. 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

6.2.2. 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, loops on the servers (if any), and 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 server starts, there will be a message on the screen indicating how long it will take the server to start.

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

To start 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 as follows:

[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 gets 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.

6.3. System Shutdown and Restart

6.3.1. System Shutdown Procedures

To shut down a machine [rhel5 or aix]:

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

sudo su -

shutdown now

6.3.2. System Restart Procedures

To restart a machine [rhel5 or aix]:

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

sudo su -

reboot

6.3.3. System Reboot Procedures

To restart a machine [rhel5 or aix]:

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

sudo su -

reboot

7. APPENDIX

7.1. Data Flow

Details of the system-level data flow can be found in the SWA of this project.

7.2. Input Data Files

Six types of VIIRS granule files including geolocation (GITCO), reflectances at top of atmosphere (SVI01 (red), SVI02 (infrared) in HDF5 format and , Aerosol Optical Depth (AOD), Cloud Mask, and Surface Reflectance (SR) in netCDF4 format Besides, ancillary Data Files are as follows:

7.2.1. Water Mask

Water mask in 200 HDF5 files.

7.2.2. NVPS Climatology

Two climatology files are required by the system. They are provided for regional NVPS and global NVPS respectively.

7.2.3. Retrieval System Files

There are a number of static retrieval system files, which serve as inputs to the NVPS retrieval. Unlike input data files, retrieval system files are static and are only updated for each operational algorithm delivery. .

7.3. Intermediate Data Files

Three types of NVPS system-generated intermediate files are needed, and they include gridded daily surface reflectance, weekly EVI, and smoothed weekly EVI from phase-one smoothing.

7.4. Output Data Files

7.4.1. Quality Monitoring Files

A NVPS statistics text for a few selected sites and color-code NVPS value image are created for both the local scale NVPS and the global scale NVPS. The statistics is used to monitor the continuity of the NVPS values, and the image can be used to find data gaps spatially.

7.5. Archive Data Files

Two NVPS files have been proposed for archive each day – a regional NVPS NetCDF file and a global NVPS NetCDF file.

8. List of References

NDE (2007), NDE Critical Design Review (NDE_CDR-20070918).

NDE (2008a), NDE Product Generation Application Interface, Version 0.4.

NDE (2013), Standards for Algorithm Delivery and Integration Using Delivered Algorithm Packages (DAPs), Version 1.4

NDE (2017), NDE Operations Handbook Version 1.0

NPOESS (2009), NPOESS Common Data Format Control Book, Version D

NESDIS/STAR (2014), GVF Algorithm Theoretical Basis Document, Version 1.0

NESDIS/STAR (2014), GVF Software Architecture Document, Version 1.0

NESDIS/STAR (2014), GVF Test Readiness Document (TRD)

NESDIS/STAR (2014), GVF External User Manual, Version 1.0

NESDIS/STAR (2014), GVF Requirements Allocation Document, Version 1.0