

Amazon Web Services (AWS) Cloud Platform for Satellite Data Processing

Richard Baker, Peter MacHarrie, Hieu Phung, Jonathan Hansford, Jakku Reddy, Stephen Causey, John Sobanski,
Steven Walsh, Ronald Niemann, Daniel Beall

Solers, Inc.

7474 Greenway Center Dr., Suite 400, Greenbelt, MD 20770; (240) 790-3338

richard.baker@solers.com

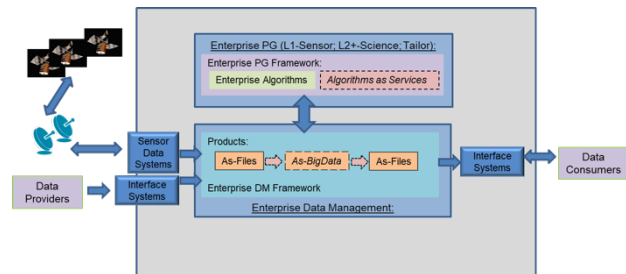
ABSTRACT

As part of NOAA's Environmental Satellite Processing and Distribution System (ESPDS) program, Solers created a cloud platform for satellite data management and processing. It consists of Enterprise Data Management (EDM) and Enterprise Product Generation (EPG) services, hosted in an Amazon Web Services (AWS) cloud environment, leveraging AWS cloud services and existing NOAA product generation algorithms. While this cloud platform was developed in the context of NOAA/NESDIS satellite data management and processing requirements, it also has tremendous applicability and cost effectiveness for small satellite data management and processing needs. An attractive method for ingesting data from small satellites is the AWS Ground Station. This can help small satellite operators save on costs of real estate, hardware/software, and labor to deploy and operate their own ground stations. The data is ingested via AWS-managed antennas, and made available for further processing in the AWS cloud using COTS RF/ baseband over IP transport services. Once this data has been ingested and made available, the flexible REST APIs from the EDM and EPG services in the AWS cloud make it easy and cost-effective for small satellite operators to catalog and process the data into consumable products, and make them available for access to end users.

INTRODUCTION

NDE Proving Ground (NPG)

The NOAA Data Exploitation (NDE) Proving Ground (NPG) is a public cloud prototype hosted in the Amazon Web Services (AWS) cloud that implements recommendations from the National Oceanic and Atmospheric Administration (NOAA) / National Environmental Satellite, Data, and Information Service (NESDIS) Office of Satellite Ground Services (OSGS) study, *Enhanced/Enterprise Product Generation Framework Study and Analysis Report*.¹ The study focused on a proposed architecture using EDM and EPG components capable of generating both level one (L1) and higher products and of integrating all products from multiple ground systems. The study proposed that L0/L1 producers (Sensor Data Systems) within the NESDIS Enterprise deliver data directly to Enterprise Data Management. An Interface Systems component accepts and delivers data to systems that are external to the "NESDIS enterprise."



**Figure 1: NOAA/NESDIS OSGS
Enhanced/Enterprise Product Generation Study
Proposed Architecture**

The NPG project has developed an AWS cloud-native EDM/EPG prototype that is executing all of NOAA's operational algorithms currently available in the on-premise operational ESPDS NDE system, and is providing web services interfaces to operational equivalent GOES-16, GCOM-W, and JPSS products.

The project has developed and studied the feasibility and cost of an operations-scale EDM/EPG system that integrates and executes the current (January 2019) NESDIS operational product release technical baseline (PRT), in an AWS cloud environment.

Development of the NPG started in January 2018. The project established an AWS cloud environment in February 2018. EDM and EPG service development completed in February 2019.

The NPG ran a 21 day operational load scenario in March 2019.

AWS Ground Station?

AWS Ground Station is a fully managed AWS cloud service (“Ground Station as a Service”) that allows satellite operators to control satellite communications, process data, and scale operations without building or managing their own ground station infrastructure. AWS Ground Station provides a fully managed network of ground station antennas located around the world in close proximity to AWS infrastructure regions. Satellites are used for a wide variety of use cases, including weather forecasting, surface imaging, communications, and video broadcasts. Ground stations form the core of global satellite networks. These facilities provide communications between the ground and the satellites in space. Today, satellite operators must either build their own ground stations, or obtain long-term contracts with ground station providers, often in multiple countries to provide enough opportunities to contact the satellites as they orbit the globe. Once all the data is downloaded, a variety of additional infrastructure including servers, storage, and networking in close proximity to the antennas are required in order to process, store, and transport the data from the satellites.

AWS Ground Station provides direct access to other AWS services and the AWS Global Infrastructure including a low-latency global fiber network located where the satellite data is downlinked into the AWS Ground Station. This enables satellite operators to easily control satellite communications, quickly ingest and process the satellite data, and rapidly integrate that data with other applications and other services that are running in the AWS cloud. For example, satellite operators can use AWS S3 to store the downloaded data, AWS Kinesis Data Streams for managing data ingestion from satellites, and AWS SageMaker for building custom machine learning applications that apply to the satellite data sets. AWS Ground Station can help satellite operators save up to 80% on the cost of ground station operations by allowing them to pay only for the actual antenna time used, and relying on the AWS global footprint of ground stations to download data when and where they need it, instead of building and operating their own global ground station infrastructure. There are no long-term commitments, and satellite operators gain the ability to rapidly scale their satellite communications on-demand as required.

NPG SYSTEM DESCRIPTION

The NPG is an AWS cloud hosted, operations-scale EDM/EPG system that is compatible with existing

NESDIS operational product generation capabilities provided by NOAA’s on-premise operational ESPDS NDE system. It provides all existing interfaces that support algorithm and product integration while providing several additional enhancements.

NPG Capabilities

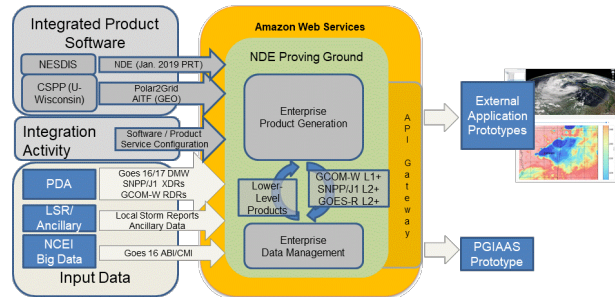


Figure 2: NDE Proving Ground (NPG)

The NPG integrates product software (i.e., Science Algorithms) and Ground System provided inputs to generate all current (as of March 2019) products that are currently being generated by NOAA’s on-premise operational ESPDS NDE system. The NPG also provides substantial scaling capabilities and application interfaces that provide access to products and to on-demand job services. A Python client library has been developed to assist with utilization of the APIs. Example applications have been built to demonstrate the EPG and EDM interfaces.

NPG Architecture

The NPG is composed of both custom-developed application and cloud provider services. Its major components are EDM, EPG, and Data Transport. AWS provided services are used extensively by each major component.

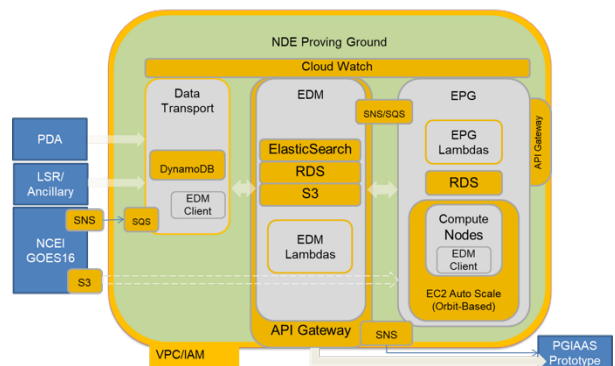


Figure 3: NPG High Level Architecture

The NPG integrates custom application software developed during the prototype project with AWS cloud services to implement ESPDS-compatible Data and Product Generation services in a public cloud environment. The NPG is fully compatible with the existing Product Integration capabilities of ESPDS NDE and the NESDIS Delivered Algorithm Package (DAP) standard for delivery of NESDIS Product Algorithm Software. The current NPG is configured to ingest data from NOAA’s on-premise ESPDS Product Distribution and Access (PDA) Integration & Test (I&T) Environment at the NOAA Satellite Operations Facility (NSOF) in Suitland, MD; the NOAA National Centers for Environmental Information (NCEI) Big Data Project’s AWS S3 bucket; and various ancillary and other data providers.

While the NPG is compatible with NOAA’s on-premise operational ESPDS NDE system, it extends the current ESPDS NDE software integration, data access, and product generation capabilities with a RESTful API implementation.

The NPG uses the AWS cloud services identified in Table 1.

Table 1: NPG AWS Cloud Services

AWS Cloud Service	Function
API Gateway	Scalable RESTful APIs integrated with backend services
CloudWatch	Monitoring, logging, and management services
DynamoDB	Scalable noSQL database
Elastic Block Store (EBS)	Persistent block storage for EC2 instances
Elastic Cloud Compute (EC2)	Compute resources
Elastic File System (EFS)	Shared Linux-based, elastic file system
Elasticsearch Service	Scalable full text (metadata) search
Relational Database Service (RDS) - PostgreSQL	Managed PostgreSQL RDBMS
Simple Notification Service (SNS)	Publish/subscribe messaging service
Simple Queue Service (SQS)	Message queuing service
Simple Storage Service (S3)	Global object storage
Virtual Private Cloud (VPC)	Logical isolation of privately networked components
Cloud Formation	Cloud service resource description and provisioning. (i.e., “infrastructure as code”)
Identity and Access Management (IAM)	Authorization and access management for both users and services
Lambda	“Serverless” Function as a Service (FaaS) application deployment and run-time environment
Cloud9	Cloud-based integrated development environment (IDE)

The major components of the NPG are Enterprise Product Generation (EPG), Enterprise Data Management (EDM) and Data Transport.

NPG Enterprise Product Generation (EPG)

EPG is composed of the following components: product software integration and work flow configuration, job scheduling (factory), and resource and workload management. Additionally, the EPG provides on-demand APIs that can be used for various product generation scenarios.

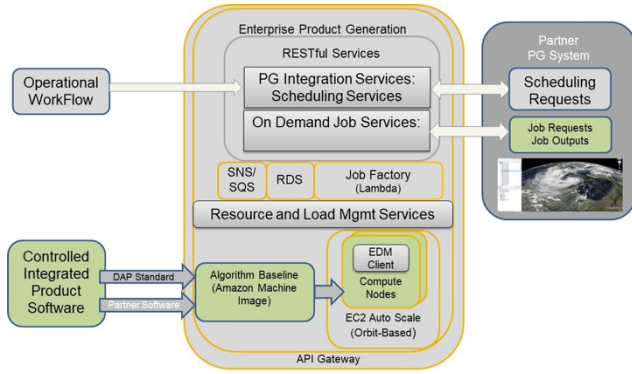


Figure 4: NPG Enterprise Product Generation (EPG) High Level Architecture

The EPG component provides the following capabilities.

ESPDS NDE Compatible Capabilities:

- Product Software Integration
- Job Factory Service (Job Creation and Queuing)
- Run-time environment, load and resource management (Scaling)

Extended Services:

- Integration and Scheduling for External Product Generation Systems
- On-Demand Job and Tailoring Invocation

EPG Product Software Integration

The EPG component provides a facility for integrating product software (algorithms) with other NPG components. The software is integrated using Amazon Machine Images (AMI) within AWS EC2. The workflow of the product software is implemented using JSON specifications via RESTful web service API calls.

EPG Job Factory

The EPG Job Factory responds to the arrival of inputs by creating an incomplete job specification using production rules created by the product integration services. The EPG Job Factory monitors incomplete job specifications and the EDM’s data holdings to determine when/if a job specification has completed and whether to enqueue a production job.

EPG Load and Resource Management

The EPG component distributes production jobs to computation and storage resources based upon the job priority (KPP, High, Medium, and Low) and job class

(Small, Medium, Large, etc.) values set in a job’s production rule.

The EPG component utilizes AWS EC2 services to manage the resource necessary to meet work load requirements. The basic workload components are:

- AMI – Virtual machine (VM) templates containing operating system (OS), product software, node manager, boot and shutdown scripts, initial storage configuration.
- Launch Configuration – Maps AMI to an AWS EC2 Instance Type. The instance types provide compute, input/output (I/O), memory, and network capacity of the desired compute node.
- Auto Scale Group – Identifies how many AWS EC2 instances to run at any given time.
- EPG Compute Node – Active instance of an AWS EC2 EPG compute node.

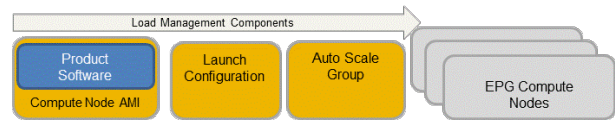


Figure 5: EPG AWS Scaling Service Components

The NPG project developed multiple types of EPG load management:

- Static Configuration
- Auto Scale by Orbit Schedule
- Auto Scale Under Backlog

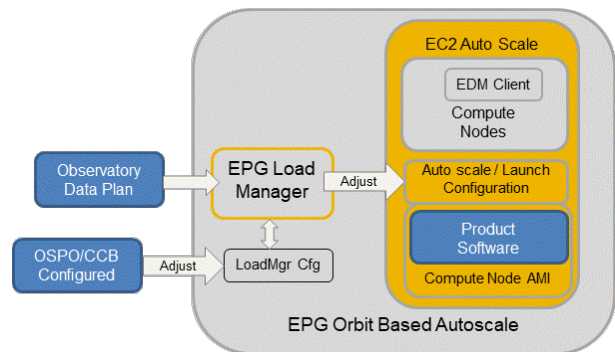


Figure 6: EPG Orbit-Based Auto Scaling

EPG job prioritization is implemented using AWS SNS and AWS SQS to route and prioritize jobs to EPG compute resources. The configuration provides preference to NOAA’s key performance products (KPPs) and products with shorter latency requirements over products that have longer latency requirements.

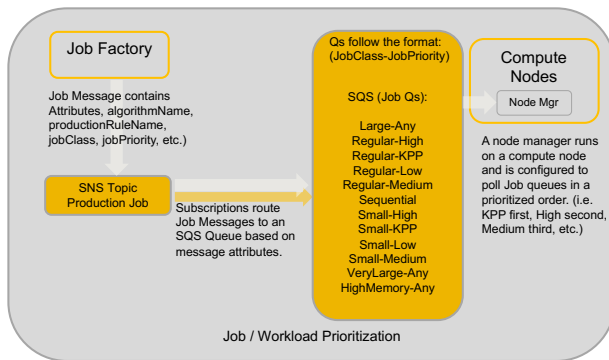


Figure 7: EPG Job and Workload Management

EPG Extended PG Services

The EPG APIs provides access to algorithm/workflow registration and on-demand job invocation services. The NPG project developed an example GUI to demonstrate the capability. The GUI requests an on-demand job which matches GOES Mesoscale products with Local Storm Reports and returns an animated output product.

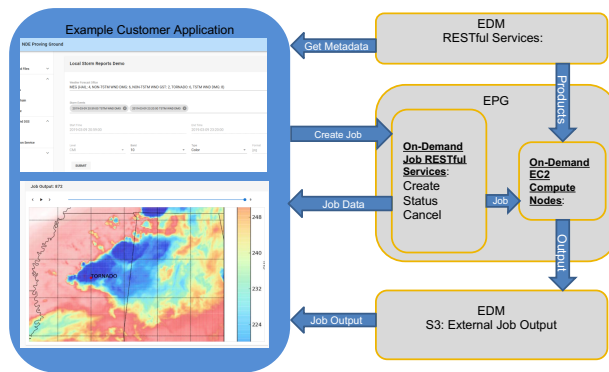


Figure 8: NPG Example EPG Job Service API Utilization

EPG Federated Product Generation

The NPG project integrated the University of Wisconsin Space Science and Engineering Center (SSEC)’s Community Satellite Processing Package (CSPP) Geo Algorithm Integration Team Framework (AITF) software package to demonstrate the capability of running an algorithm in a federated product generation mode utilizing AWS SNS subscriptions.

The CSPP Geo package processes Level 1B data from the Advanced Baseline Imager (ABI) instrument on NOAA’s GOES-16 satellite, and creates Level 2 derived geophysical products. The CSPP Geo software produces output files in netCDF4 format and also

contains a program to create “quicklook” images, plots of select variables from the netCDF4 output files in PNG format.

The CSPP Geo package was provided as a set of pre-compiled binaries intended to run on CentOS 6. The NPG project developed a driver script to launch the algorithm in a manner consistent with the DAP standards, along with the necessary suite of product, algorithm, and production rule definitions. The production rule definitions were assigned a “Boutique” job class, which was used to route the EPG Production Job AWS SNS messages related to this algorithm to separate AWS SQS queues from the ones used by the rest of the enterprise, allowing the CSPP Geo algorithm to be run on AWS EC2 instances dedicated to it. A Python script separate from the aforementioned driver script was written to handle de-queuing jobs from those AWS SQS queues as they became available, obtaining the input files from the inventory, and making the output available to ingest.

EPG Services API

Both EPG and EDM use an AWS API Gateway as the entry point into the system from end clients. The AWS API Gateway sits between the client and the system and is responsible for request routing, composition, and protocol translation. Without an AWS API Gateway, clients would need to send requests directly to different services, which can lead to tight coupling, extraneous requests, and security issues.

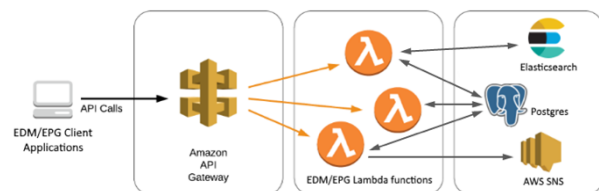


Figure 9: NPG APIs Services Stack

In the NPG, the AWS API Gateway invokes AWS Lambda functions, passing the request parameters from the clients. The AWS Lambda functions in the NPG are written in Python and they interact with AWS RDS, AWS Elasticsearch Service, and other AWS services (such as AWS SNS) to fulfill the clients’ requests.

The EPG Services API provides access to Algorithm, Production Rule, and Production Job related functions by exposing the following operations to end clients.

Table 2: EPG Services API

Operation	Description
EPG Product Software Integration Service API	
Register/Update Algorithm	Registers a new Algorithm definition in both AWS RDS and AWS Elasticsearch Service. If the Algorithm definition exists, it will be updated.
Get Algorithm List	Returns a list of all the registered Algorithms with basic information for each one: id, name, version, type.
Get Algorithm	Returns the full details for a specified Algorithm. These details are equivalent to the input of Register/Update Algorithm.
Register/Update Production Rule	Registers a new Production Rule definition in both AWS RDS and AWS Elasticsearch Service. If the rule definition exists, it will be updated.
Get Production Rule List	Returns a list of all the registered Production Rules with basic information for each one: id, name, type, active flag.
Get Production Rule	Returns the full details for a specified Production Rule. These details are equivalent to the input of Register/Update Production Rule.
Toggle Production Rule	Updates the PR Active Flag attribute for a Production Rule.
EPG On-Demand Job Service API	
Create Job	Creates an on-demand Production Job (including Job Specification).
Get Jobs Summary	Returns a list of on-demand Production Jobs with basic information, grouped by Algorithm name and by Job Status.
Get Job Details	Returns the current details for an on-demand Production Job, including its status and outputs (when the Job is COMPLETE).
Update Job	Updates the attributes of a specific on-demand Production Job (currently only job Status).
Search Job	Allows for searching of on-demand Production Jobs. Query parameters can include job Status, algorithm, enqueueTime, startTime, completionTime. Optionally, results can be sorted or limited.

Table 3: EPG Services API CRUD Matrix

Operation	Create	Read	Update
EPG Product Software Integration Service API			
Register/Update Algorithm	X		X
Get Algorithm List		X	
Get Algorithm		X	
Register/Update Production	X		X

Rule			
Get Production Rule List		X	
Get Production Rule		X	
Toggle Production Rule			X
EPG On-Demand Job Service API			
Create Job	X		
Get Jobs Summary		X	
Get Job Details		X	
Update Job			X
Search Job		X	

* Delete operations through the EPG Services API were not essential for the NPG.

NPG Enterprise Data Management (EDM)

EDM is composed of the following components: product integration services, data catalog and storage services, data access services. A primary goal of the EDM is to extend some of the data access capabilities currently available only within NOAA’s on-premise operational ESPDS NDE system across missions.

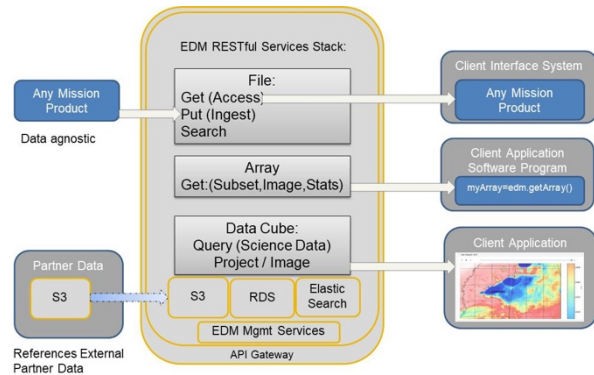


Figure 10: EDM High Level Architecture

EDM Product Integration

The EDM provides RESTful services that define products and the product’s data structures for use by other EDM services. For example, data product definitions are used by the EPG services to describe algorithm inputs and outputs.

EDM Ingest

The EDM Ingest service provides a single RESTful interface for product catalog, storage, and notification services. The EDM Ingest service:

- Validates and processes incoming data via AWS Lambda functions and Docker containers (NOTE: some ingested files exceed the storage capacity of AWS Lambda functions).

- Stores product data (i.e., HDF5, netCDF, BUFR, etc. files) in AWS S3.
- Extracts and stores metadata in AWS Elasticsearch Service and in AWS RDS.
- Provides a notification of new data using AWS SNS.

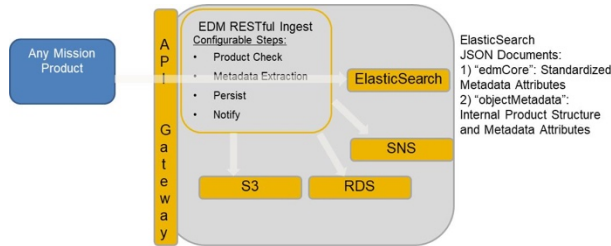


Figure 11: EDM Ingest Service

The EDM Ingest service extracts and makes available, all internal product metadata for selected formats. (e.g., HDF5, netCDF4). The metadata is converted to a JSON document containing two major divisions:

- EDM Core Metadata: Metadata attributes common among all mission products.
- Object Metadata: Metadata attributes specific to the file’s internal structure and content.

The metadata derived by the EDM Ingest service provides information about the internal structures (e.g., dimensions and arrays) of the products managed by the EDM, and this information is used by the EDM services as described below.

EDM Data Access

EDM provides several data access capabilities: file search and access, internal array content access (i.e., subsetting by direct resource path), and data selection service query.

The NPG has developed a GUI application to demonstrate EDM data access capabilities.

EDM Search

Full text, metadata attribute, and spatial search capabilities are provided in a public RESTful API across the EDM data holdings. An example search and data access GUI (application) is part of the prototype.

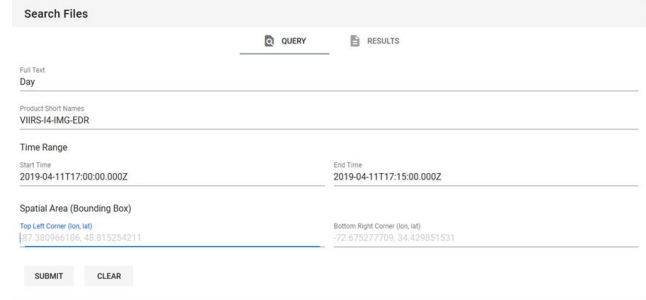



Figure 12: EDM Search API Demonstration GUI

The search API can accept a wide variety of search inputs, and provides several levels of detail and sort options. EDM Search uses AWS Elasticsearch Service for field and full-text searches and AWS RDS for spatial searches, and manages the combination of search results between the two backend databases.

EDM File Access

In addition to metadata content, the science data contents of EDM are accessible via its RESTful API. Data is accessible at multiple levels including File, Array, and Data Cube.

An example of the File access level is the  icon on the EDM File Access Demonstration GUI. Clicking the icon will download the file via an HTTPS GET request to the file resource in AWS S3.

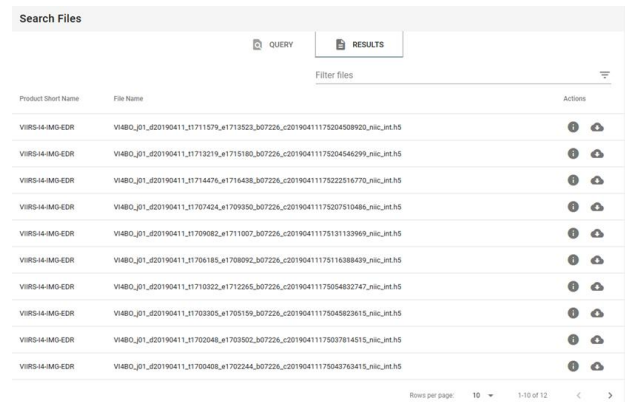


Figure 13: EDM File Access Demonstration GUI

EDM Data Access intermediates between the external data consumer’s system and EDM’s isolated S3 bucket. When accessed, an EDM file’s URL, creates and then returns a signed certificate to the file’s S3 bucket address with an http redirect directive to the requesting client. The signed certificate is valid for a limited time period and the data is obtained using the temporary redirected URL.

EDM Array Access

Internal arrays of data/metadata within individual files are accessible via EDM's /file/array resource URL. This interface invokes an AWS Lambda function that extracts the requested internal array from the file and returns the array as a binary file or as a byte stream. Figure 14 shows an example of extracting of the BrightnessTemperature array from a VIIRS-I4-IMG-EDR file listed in Figure 13.



Figure 14: EDM Array (aka Subset) Access API Demonstration GUI

The EDM Array Access API can additionally be used to load the EDM array into a client application program's memory using the EDM Client library.

EDM Data Cube Access

EDM provides an aggregated view (known as a "data cube") of science data that shares the same geo-temporal footprints. The data cubes can be queried to provide common tailoring services. Figure 15 provides an example of aggregation, re-projection, and imaging of VIIRS radiances. The data selection service provides subsetting, subsampling, filtering, aggregation, re-projection, and formatting of science data. Figure 15 shows a limited sample interface to the EDM Data Cube Access API. Results of the sample query are shown in Figure 16. The EDM Data Cube Access query is submitted as an on-demand job to the EPG services. Like a file's URL, job outputs are accessible via a signed-certificate to the job's output location within an AWS S3 bucket.

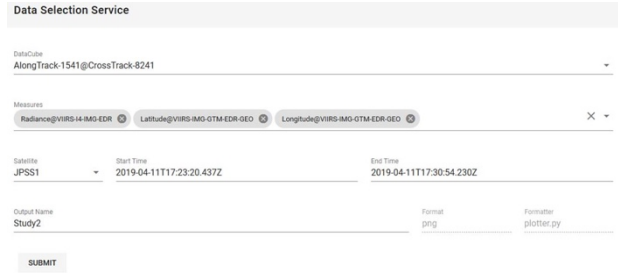


Figure 15: EDM Data Cube Access API Demonstration GUI

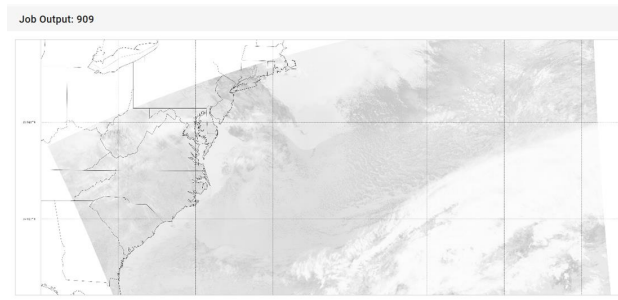


Figure 16: EDM Data Cube Access API View Output Demonstration GUI

The image in Figure 16 is a project image of data from the VIIRS-I4-IMG-EDR product from NOAA's JPSS-1 satellite captured on 11 April 2019 17:23:20Z – 17:30:54Z.

EDM Services API

The EDM Services API provides access to Product, File, and Data Cube related functions by exposing the following operations.

Table 4: EDM Services API

Operation	Description
EDM Product Software Integration Service API	
Register/Update Product	Registers a new Product definition in both AWS RDS and AWS Elasticsearch Service. If the Product definition exists, it will be updated.
Get Product List	Returns a list of all the registered Product with basic information for each one: id, short name, description, platform
Get Product	Returns the full details for a specified Product. These details are equivalent to the input of Register/Update Product.
Search Product	Searches for Products given a productFilenamePattern.
EDM Data Ingest Service API	
Ingest File	Given an AWS S3 bucket and key referring to a file, ingests a file by performing validation, metadata extraction, and catalog of the file.
EDM Data Access Service API	
Get File	Sends a requested file to the client via an AWS S3 pre-signed URL.
Search File	Allows for searching of files. Query parameters can include full Text, productShortNames, fileIds, fileNames, timeRange (startTime and endTime), spatialArea, orbitRange (beginOrbit and endOrbit number), fileDayNightFlag, fileAscDescIndicator, and any other internal group attributes. Optionally, result fields can be specified and results can be sorted or limited.
Get File Metadata	Returns a JSON string of a file's internal metadata structure.
Get File Array	Returns the specified binary data (array) of a file via an AWS S3 pre-signed URL.
Get Data Cube List	Returns a listing of all data cubes with information for each one: name, productShortNames included in the data cube, count of the measures represented by the data cube.
Get Data Cube Metadata	Returns the metadata content for a specified data cube's structure: id, name, dimensions, measures
Get Data Cube Select	Invokes the EDM Data Cube Access Service given a "select" query. EDM will create a Data Cube Access job that will create the output (i.e., netCDF4, PNG, etc.) specified in the query. The output can be retrieved using the EPG Get Job Details API.

Table 5: EDM Services API CRUD Matrix

Operation	Create	Read	Update
EDM Product Software Integration Service API			

Register/Update Product	X		X
Get Product List		X	
Get Product		X	
Search Product		X	
EDM Data Ingest Service API			
Ingest File	X		
EDM Data Access Service API			
Get File		X	
Search File		X	
Get File Metadata		X	
Get File Array		X	
Get Data Cube List		X	
Get Data Cube Metadata		X	
Get Data Cube Select	X		

* Delete operations through the EDM Services API were not essential for the NPG.

NPG Data Transport

Data transport provides external interface services to data providers external to the NPG. The NPG project implemented the following interfaces for the prototype:

- NOAA's On-Premise ESPDS PDA I&T System: Continuous polling and pulling of Suomi National Polar Partnership (S-NPP), JPSS-1, and Global Change Observation Mission – Water (GCOM-W) satellite data files via FTPS.
- NOAA's NCEI Big Data Project AWS S3 Bucket: AWS SNS notifications, and pulling of GOES-16 satellite data files via HTTPS.
- Ancillary and Other Data Providers: Periodic polling and pulling of ancillary and other data files needed for product generation via FTP, FTPS, and HTTPS.

NPG API Client Libraries

The NPG project has developed a Python client library that encapsulates the EPG and EDM RESTful APIs into more easily understandable function calls. A tutorial of the client libraries is available at: https://jupyterhub.ndepg-jupyter.com/user/nde/notebooks/Use_Case_Scenarios/.

NPG Monitoring

Monitoring of the NPG is accomplished using the AWS CloudWatch service. AWS CloudWatch provides metrics collection and reporting of most AWS services. An AWS CloudWatch graphing example is show in Figure 17.

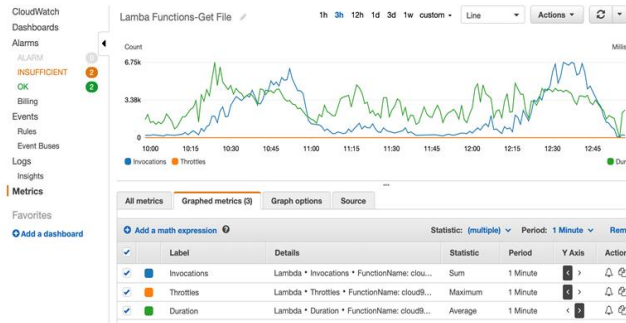


Figure 17: NPG Monitoring and Graphing in AWS CloudWatch

Additionally, NPG uses AWS CloudWatch to monitor application events and metrics related to data ingest and product generation, as shown in Figure 18.

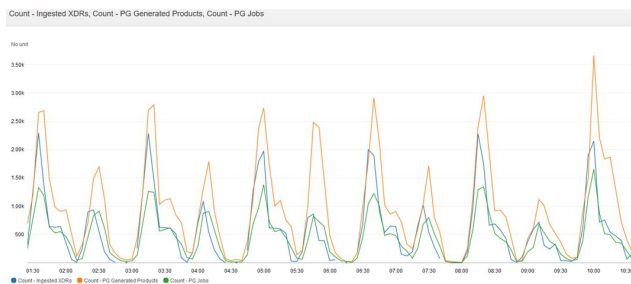


Figure 18: NPG Monitoring of Application Events and Metrics in AWS CloudWatch

NPG Design and Implementation Trades

The following trades occurred during the development phase of the NPG project.

Data Transport Interface Implementation

While the Data Transport capability of pulling data from partner AWS S3 buckets is desirable, (NOAA’s NCEI GOES-16 and pending JPSS CGS), pulling data from NOAA’s on-premise ESPDS PDA system is a sub-optimal approach. A preferred approach is for NOAA’s on-premise ESPDS PDA system to push data directly to the NPG’s AWS S3 bucket. However, that capability would have required a change to NOAA’s on-premise ESPDS PDA system’s baseline. That change was considered undesirable until the outcome of the study is understood.

Distribution was not included within the domain of the NPG project’s area of study.

Scaling and Interoperability Implementation

One of the initial purposes of the NPG project was to evaluate Red Hat JBoss Fuse as a replacement to Red

Hat JBoss SOA-P, which is used by NOAA’s on-premise operational ESPDS NDE system for scaling and interoperability, and has reached end of life. Initial versions of the NPG prototype were implemented with Red Hat JBoss Fuse. However, during the NPG prototype effort, it became understood that the Red Hat JBoss Fuse licensing policy was incompatible with the product’s use on the EPG compute nodes (priced per core). The bulk of the compute power on EPG compute nodes is utilized by algorithms, not by the Red Hat JBoss Fuse node manager. An implementation of the node manager functionality using Python was developed and appears to be a viable, lighter weight, and lower cost alternative.

Red Hat JBoss Fuse was also used for initial Data Transport, EDM Ingest, and EPG Job Factory implementations. Data Transport has since been replaced by a Python implementation. EDM Ingest and EPG Job Factory have been refactored to use AWS services, which provide the necessary scaling and interoperability capabilities of Red Hat JBoss Fuse, without the additional licensing cost.

RESTful APIs Implementation

The RESTful APIs of the NPG could have been implemented using a vast number of products. However, the ability to implement an interface as reliable, scalable, and available as the AWS API Gateway, given the project’s limited schedule and resources, would have been difficult to accomplish.

A noteworthy aspect of the AWS API Gateway is that it forces a separation of the API’s implementation from the back-end services. In general, during software development, there is often a drift of front-end API logic to the back-end service and of the service to the front-end API logic that complicates RESTful service development and maintenance. Forcing a separation of the API’s implementation from the back-end services helps to prevent these complications.

NoSQL Database Implementation

AWS Elasticsearch Service was available as a managed cloud service at the beginning of the prototype and was selected primarily for that reason. AWS subsequently released a compatible MongoDB service in January 2019, which has not been evaluated as part of the NPG project.

The NoSQL databases have been found to have a specific disadvantage for near-real-time transaction applications. For applications that have a large number simultaneous transactions (e.g., ingest metadata updates) the transactions must be grouped into batches

to achieve effective throughput rates that keep up with latency requirements. AWS provides this type of batching service (AWS Kinesis Firehose), but the specific implementation causes an increase in ingest latency of up to one minute and requires a schema change that can impose significant changes in an application.

NPG continued with the use of AWS Elasticsearch Service, but wrote a custom batching mechanism to ensure latency requirements are met.

Algorithm Implementation

NPG implements the current form of NOAA’s product algorithms as executables, but based on containerized algorithms that were delivered on 1 April 2019, NPG could easily integrate algorithm containers.

Other implementations are possible, but not as the majority of NOAA’s product algorithms are currently developed. For example, AWS Lambda functions have been proposed as alternative implementations of algorithms. However, AWS Lambda functions are currently restricted to a maximum of 3 GB memory, 15 minutes runtime duration, 0.5 GB storage, and 250 MB deployment package. The implementation of algorithms as “micro-services” will require changes not just to the algorithms themselves but also to how data is stored and managed. This is what the NOAA/NESDIS OSGS *Enhanced/Enterprise Product Generation Framework Study and Analysis Report* referred to as “Products-as-BigData”, “Algorithms-as-Services”.¹

The containerized implementation of algorithms is highly desirable, especially for cloud-based implementations. However, the conversion of NOAA’s existing product algorithm software to containerized algorithms is a labor-intensive effort, not only with implementation but especially regarding validation of the resulting products.

The NPG project received two containerized NOAA product algorithms after completion of the 21 day test. The containerized algorithms were compared to stand-alone executable versions. The average run-time durations are provided in Table 6.

Table 6: Executable vs. Containerized NOAA Product Algorithm Run Times

Unit Name	Executable Mean Runtime (seconds)	Container Mean Runtime (seconds)	Difference %
VPW PRODUCT JPSS1	241.03	243.99	1.23%
Cloud Mask JPSS1	258.93	267.58	3.34%
Cloud Mask NPP	256.73	270.56	5.39%
VPW REMAP NPP	131.92	146.09	10.74%
VPW PRODUCT NPP	355.31	358.02	0.76%
VPW REMAP JPSS1	128.56	139.61	8.59%

NPG Process, Performance, and Costs

Data Ingest Cycle

The NPG continuously receives all input data necessary to produce the current operational products and in response generates products at a rate that meets or exceeds current requirements. Figure 19 depicts the rate at which data that is used as input to the NPG is ingested from data providers in terms of number of files over time.

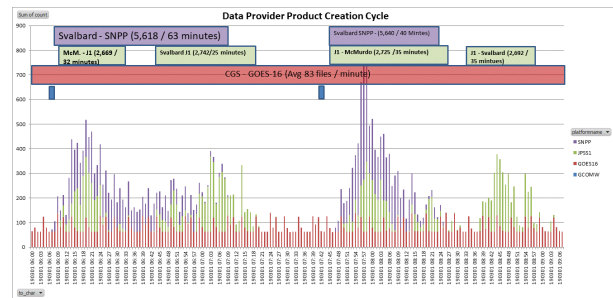


Figure 19: NPG Data Ingest Cycle

Table 7 provides the daily data ingest count and volume.

Table 7: NPG Daily Data Ingest Volume

Satellite	Ingested Data Types	Ingested Data Volume (Files per Day)	Ingested Data Volume (MB per Day)
GCOM-W	1	14	1,756
GOES-16	64	52,224	136,084
JPSS-1	65	77,756	907,447
S-NPP	73	84,256	909,847

Product Generation Cycle

In response to the arrival data, the NPG generates products using operationally delivered product software (algorithms and tailoring) in a work flow and resource configuration that exceeds existing latency requirements. Specifically, the NPG is meeting primary sensor latency requirements for both S-NPP and JPSS-1 generated products. Figure 20 depicts the rate at which the NPG generates products in terms of number of files over time.

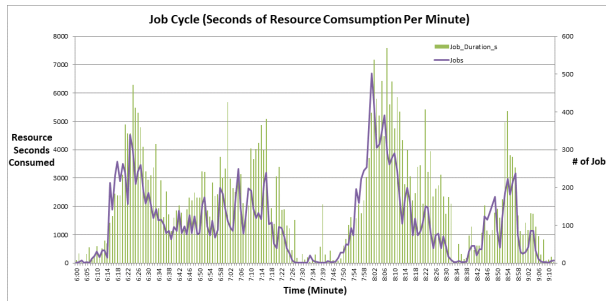


Figure 20: NPG Product Generation Cycle

Table 8 provides the NPG daily generated product count and volume.

Table 8: NPG Daily Product Generation Volume

Satellite	Generated Product Types	Generated Product Volume (Files per Day)	Generated Product Volume (MB per Day)
GCOM-W	32	457	31,892
GOES-16	240	181,877	940,301
JPSS-1	250	137,552	1,966,206
S-NPP	303	164,103	2,066,359

Table 9 provides the estimated average yearly compute and storage costs to continuously ingest data and generate products using NOAA’s product algorithms that were integrated into the NPG (operational scenario). In each case, the portion of the costs that is for storage is very low (less than 10%), with compute being the majority of the costs.

Table 9: NPG Estimated Yearly Compute and Storage Costs

Satellite	Estimated Yearly Compute and Storage Costs
GCOM-W	\$2,171.37
GOES-16	\$79,690.10
JPSS-1	\$59,483.85
S-NPP	\$59,483.85

AWS GROUND STATION SYSTEM DESCRIPTION²

AWS Ground Station enables satellite operators to control and ingest data from orbiting satellites without having to buy or build satellite ground station infrastructure. AWS Ground Station does this by integrating the ground station equipment like antennas, digitizers, and modems into the AWS regions around the world. Satellite operators can simply onboard their satellites and schedule time to communicate with them using AWS Ground Station. Satellite operators have the option of conducting all of their satellite operations within the AWS cloud, including the storing and processing of the satellite data and delivering products using AWS services, or use AWS Ground Station just to downlink the satellite data and transport it to their own processing center.

AWS Ground Station Features

Schedule Satellites and Download Data Using AWS Services

Satellite operators can use the AWS Ground Station console to identify the satellites they need to communicate with and schedule “Contacts” with each satellite, where each Contact consists of a selected satellite, start and end time, and the ground location. After scheduling their Contacts, satellite operators can launch Amazon EC2 instances to run each portion of the Contact. They can launch a Command EC2 instance to receive operational telemetry from their satellite and transmit commands up to the satellite to schedule future activities. They can also launch a Downlink EC2 instance to receive bulk mission data from their satellite. These EC2 instances will communicate with AWS Ground Station’s antenna gateway over an elastic network interface (ENI) connection in AWS VPC that exists for the duration of the contact.

Fully Managed Global Ground Station Network Integrated with AWS Global Infrastructure

AWS Ground Station antennas are located within fully managed AWS ground station locations, and are interconnected via Amazon’s low-latency, highly reliable, scalable and secure global network backbone.

Data downlinked and stored in one AWS region can be sent to other AWS regions over the global network, so it can be further processed.

Graphical AWS Ground Station Console

AWS Ground Station provides an easy to use graphical console that allows you to reserve contacts and antenna time for your satellite communications. You can review, cancel, and reschedule contact reservations up to 15 minutes prior to scheduled antenna times.

Direct Access to AWS Services

AWS Ground Station provides their satellite antennas direct access to AWS services for faster, simpler and more cost-effective storage and processing of downloaded data. This allows satellite operators to reduce data processing and analysis times for use cases like weather prediction or natural disaster imagery from hours to minutes or seconds. This also enables satellite operators to quickly create business rules and workflows to organize, structure, route the satellite data before it can be analyzed and incorporated into key applications such as imaging analysis and weather forecasting. Key AWS services include AWS EC2, AWS S3, AWS VPC, AWS Rekognition, AWS SageMaker, and AWS Kinesis Data Streams.

Supports Most Common Satellites and Communication Frequencies

AWS Ground Station antennas can connect with any satellite in Low Earth Orbit (LEO) and Medium Earth Orbit (MEO) operating in X-band and S-band frequencies, including: S-band uplink and downlink, X-band narrowband and wideband downlink.

Pay-Per-Minute Pricing

Satellite operators can schedule access to AWS Ground Station antennas on a per-minute basis and pay only for the scheduled time. They can access any antenna in the ground station network, and there are no long-term commitments.

AWS Ground Station Benefits

Ground Station as a Service

AWS Ground Station provides a global network of ground stations in close proximity to the global network of AWS infrastructure regions. With AWS Ground Station, satellite operators no longer need to worry about buying, leasing, building, scaling or managing their own satellite ground stations.

Trusted Security

Security is AWS's highest priority. Satellite operators will benefit from the AWS facility and network architecture built to meet the requirements of the most security-sensitive organizations. AWS Ground Station provides premium data security and physical security without any additional costs.

Pay-As-You-Go

With AWS Ground Station, satellite operators only pay for the actual antenna time they use. There are no long-term contracts, or hidden fees. With a single price, they can use any antenna in the global AWS Ground Station network.

Fast Data Downloads

With two antennas installed at each AWS Ground Station location, satellite operators can download satellite data to AWS Ground Station antennas worldwide and use it rapidly. With AWS Ground Station's global network of antennas, and easy and simple satellite scheduling on-demand, satellite operators can download their satellite data much faster and when they need it, without significant delays due to antenna scheduling delays and conflicts.

Immediate Data Processing

AWS Ground Station provides satellite antennas in close proximity to AWS infrastructure regions, giving satellite operators low-latency and low-cost access to AWS services to store and process the satellite data. This allows satellite operators to reduce data processing and analysis times for use cases like weather prediction or natural disaster imagery from hours to minutes or seconds.

Self-Service Scheduling

Satellite operators can easily schedule Contacts with their satellite(s) using the AWS Management Console and APIs. Once reserved, Contacts can be rescheduled or cancelled up to 15 minutes prior to start.

INTEGRATED AWS CLOUD PLATFORM FOR SATELLITE DATA PROCESSING

Using a combination of AWS Ground Station with a cloud-based EDM and EPG services framework as prototyped for Solers' NPG project, satellite operators can create a fully integrated AWS cloud platform that is capable not only of command and control, but also of downloading the satellite data into AWS services along with processing, analyzing, and generating user-consumable products based upon the satellite, without requiring any additional infrastructure or facilities. Small satellites would make a great use case for this

kind of a framework, as it would provide a very flexible, scalable, and low-cost capability to fully command/control the satellites as well as manage and process the downloaded satellite data, with no infrastructure or facilities overhead.

A very straightforward integration of these 2 capabilities would be to have AWS Ground Station download the satellite data as files into an AWS S3 bucket, and leverage AWS SNS notifications from the AWS S3 bucket to trigger a data ingest process as part of the EDM services. This AWS S3 and AWS SNS data ingest capability was successfully proven as a viable and even preferred data ingest option during the NPG prototype effort (for ingest of NOAA's GOES-16 satellite data). The NPG prototype effort proved the EDM and EPG services framework as capable of efficiently managing data and products from multiple satellites simultaneously, so a single instance of the EDM and EPG services framework for a particular satellite operator would allow them to manage the data and products for a single satellite, or for multiple satellites.

Once satellite data downloaded from AWS Ground Station is ingested into the EDM services, it will be available for search/discovery and access by trusted/authorized data consumers leveraging the RESTful EDM Services API, with the ability to control access permissions using native AWS security services/constructs such as AWS IAM roles and policies. If providing data consumers with access to the raw downloaded satellite data is all that is required by a particular satellite operator, then data ingest into the EDM services is all that would be required. However if additional processing and generation of user-consumable products leveraging the downloaded satellite data is also required by a particular satellite operator, then the EPG services can also be leveraged. The EPG services provide a flexible, scalable, and cost-efficient mechanism to integrate and execute algorithms that perform the necessary processing and product generation, fully accessible and configurable using the RESTful EPG Services API. Using this API, satellite operators and other trusted/authorized users can configure a set of scheduled production rules that will result in the routine generation of products based upon the ingest/arrival of required input data via the EDM services. They can also use this API to generate products on-demand, in order to meet ad-hoc types of product generation use cases that do not fit into a regularly scheduled scenario. All generated products will also be made available to data consumers via the RESTful EDM Services API.

Representing data processing and product generation algorithms as "micro-services" is recommended especially for "new" small satellite data and product use cases. Keeping the algorithm package as small and lightweight as possible will ease its portability and configuration management, and maximize the efficiency of managing its execution within the EPG services. Provided that the processing logic fits within the AWS-enforced restrictions, representing data processing and product generation algorithms as AWS Lambda functions would be the most efficient and cost-effective mechanism of integrating algorithms into the EPG services. In cases where this is not possible, then implementing and integrating the algorithms as containers would be the next recommended method to preserve the "micro-services" concept, maximize the algorithm's portability, and ease the algorithm's configuration management. Only in cases where an existing legacy code base or application must be used for an algorithm that cannot be easily or cost-effectively converted into an AWS Lambda function or a container, would it be recommended to integrate it as a standalone executable that runs directly on an AWS EC2 EPG compute node.

REFERENCES

1. Solers, Inc., [SE 16] Enhanced/Enterprise Product Generation Framework Study and Analysis Report, NOAA/NESDIS OSGS, 26 January 2018.
2. Amazon Web Services, Inc., AWS Ground Station, <https://aws.amazon.com/ground-station>, 2019.