

# File Upload Meta Data for DynamoDB Registry on HelioCloud

11/4/2022

Kiley Yeakel  
JHU/APL  
[Kiley.Yeakel@jhuapl.edu](mailto:Kiley.Yeakel@jhuapl.edu)

# Purpose of the DynamoDB File Registry

- While the file directories on S3 will mimic CDAWEB (or any other data provider), the key-object storage in S3 does not facilitate stepping through file directories
  - File directories do not exist in S3, but '/' in an object's key will be recognized as file hierarchy
  - For the sake of efficiency and cost, listing the contents of a bucket is not advised
  - DynamoDB provides a means of efficiently querying a bucket based on a search pattern
- Purpose of the DynamoDB is to facilitate file discovery in S3 via basic attributes (e.g., dataset name, time)
  - Higher-level search mechanisms (for example, if someone does not know which dataset they want & need to do data exploration) will be built on top of DynamoDB base query functionality and are not considered in scope at the moment
  - "Dataset name" must encompass the lowest level descriptor of a set of files → all files within a "dataset" contain the same data at different time steps

*Example of file hierarchy of MMS data in HelioCloud bucket, which mimics CDAWEB*

Name	Size
▼ MMS	
▼ mms1	
▼ fgm	
▼ srvy	
▼ I2	
▼ 2020	
▼ 05	
mms1_fgm_srvy_I2_20200501_v5.241.0.cdf	90 MB
mms1_fgm_srvy_I2_20200502_v5.243.0.cdf	86.92 MB
mms1_fgm_srvy_I2_20200503_v5.241.0.cdf	92.48 MB
mms1_fgm_srvy_I2_20200504_v5.241.0.cdf	91.46 MB
mms1_fgm_srvy_I2_20200505_v5.243.0.cdf	87.84 MB
mms1_fgm_srvy_I2_20200506_v5.242.0.cdf	90.36 MB
mms1_fgm_srvy_I2_20200507_v5.242.0.cdf	92.83 MB
mms1_fgm_srvy_I2_20200508_v5.242.0.cdf	91.55 MB
mms1_fgm_srvy_I2_20200509_v5.244.0.cdf	86.02 MB
mms1_fgm_srvy_I2_20200510_v5.242.0.cdf	92.7 MB
mms1_fgm_srvy_I2_20200511_v5.242.0.cdf	92.42 MB
mms1_fgm_srvy_I2_20200512_v5.244.0.cdf	87.83 MB

# Requirements for Data Upload

## 1. Intuitive File Hierarchy

- Example from CDAWEB will be provided on the next slide
- S3 buckets do not have a “directory” structure but “/” characters in the S3 filekey will be interpreted by AWS as hierarchy
- Facilitates file discovery → we aim to replicate file hierarchies used in existing archives so end-users do not have issues with data access

## 2. File Manifest

- CSV list of files to be uploaded along with accompanying meta data information
- Data can be uploaded from an external server (e.g. CDAWEB) or a S3 bucket transfer (in dev.)

## 3. Meta-data embedded within the file

- Upload pipeline will open each individual file and scrape meta data that is not provided by the manifest

***Both the file manifest and embedded meta-data enables registry in AWS DynamoDB dataframes, which will be used by end users for file discovery***

# Intuitive File Hierarchy

- Files must be organized as they would like to be presented in S3 buckets
- Each file must have a unique filepath → this will be the S3 key
  - Filepath must contain a timestamp and version number (if wanted), as well as a “product name”
    - Product name should describe the lowest-level of data, further division is just by time
  - Suggestions to mimic the MMS hierarchy if possible:

Name	Size
✓ MMS Mission	
✓ mms1 Spacecraft (not needed for single spacecraft)	
✓ fgm Instrument	
✓ srvy Instrument Mode (high rate, low rate, typically refers to sample cadence)	
✓ I2 Level of processing (I1/I2/I3)	
✓ 2020 Time differentiators (not necessary but reflects CDAWEB)	
✓ 05	
mms1_fgm_srvy_I2_20200501_v5.241.0.cdf	90 MB
mms1_fgm_srvy_I2_20200502_v5.243.0.cdf	86.92 MB

**Embed the timestamp & file version in the filename**

# Upload File Manifest

- Upload pipeline is triggered by csv file uploaded to a specific S3 bucket
- Each line of the CSV file contains meta-data about a specific file upload request:
  - **s3\_key**: where the file will live in the s3 bucket (should be similar to the upload path without the prefix)
  - **s3\_bucket**: which bucket on GSFC account the data will be uploaded to (GSFC can provide this)
  - **upload\_path**: this can either be the download link for data residing on a public server, or a S3 bucket within or outside the GSFC account
  - **product**: the dataset descriptor
  - **source\_update**: when the data was last updated by the source
  - **mission**: (e.g. MMS)
  - **sc**: spacecraft identifier, not applicable for missions with a singular spacecraft (e.g. MMS1)
  - **instr**: instrument (e.g. Flux gate magnetometer = FGM)
  - **instr\_mode**: mode the instrument was in for the data collection (not required)
  - **level\_proc**: level of processing for the dataset (L1, L2, L3, L4, etc.)

# File Meta Data

- Data upload pipeline will open each file and scrape meta data contents
- Desired meta data contents (based off CDF files)
  - zVariables: a list of strings of all the variables present in the file (will be registered in the dataSummary DynamoDB)
  - Epoch: the time strings for the entire file, need to be able to provide a start and end date for each file if applicable
  - Version: the version number of the file
- If you would like an additional query pattern for your dataset, e.g., wavelength, exposure time, etc, it must be included as either meta data within the file itself (safest option) or within the file manifest




# fileRegistry DynamoDB

The lowest level file identifier, enabling simple queries on meta-data

- Primary key: s3\_filekey
- Secondary query patterns:
  - Dataset\_datesort: composite key of dataset (partition) and start date of file (sort)
    - Retrieves files belonging to a particular dataset and time range, sorted by time
  - Dataset\_uploadsort: same as above but date is when the file was uploaded to AWS
  - Instr\_id: sparse index of just the instrument identifier
  - Level\_id: same as instr\_id but for level of processing
  - Mission\_id: same as instr\_id but for mission+sc identifier

fileRegister

 Autopreview

View table details

▼ Scan/Query items

Scan/query a table or index

Scan

Query

dataset\_uploadsort ▲

dataset (Partition key)

Table

Enter partition key value

fileRegister

date\_upload (Sort key)

Index

Equal to ▼

Enter sort

dataset\_uploadsort ✓

instr\_id

dataset\_datesort

level\_id

mission\_id

► Filters

Run

Reset

s3_filekey ▼	dataset ▼	dataset_update ▼	end_date ▼	instrument ▼	level_processing ▼	mission ▼	source_update ▼	start_date ▼	version
<a href="#">mms/mms1/fgm/srv...</a>	mms_mms1_fgm_srvy_l2	2022-11-04T18:4...	2020-02-1...	fgm	l2	mms_mms1	2020-03-23T00:...	2020-02-16...	3.6.3
<a href="#">mms/mms1/fgm/srv...</a>	mms_mms1_fgm_srvy_l2	2022-11-04T18:4...	2020-03-1...	fgm	l2	mms_mms1	2020-04-13T02:...	2020-03-10...	3.6.3
<a href="#">mms/mms1/fgm/srv...</a>	mms_mms1_fgm_srvy_l2	2022-11-04T18:4...	2020-02-1...	fgm	l2	mms_mms1	2020-03-23T00:...	2020-02-15...	3.6.3
<a href="#">mms/mms1/fgm/srv...</a>	mms_mms1_fgm_srvy_l2	2022-11-04T18:4...	2020-03-0...	fgm	l2	mms_mms1	2020-04-03T00:...	2020-02-29...	3.6.3
<a href="#">mms/mms1/fgm/srv...</a>	mms_mms1_fgm_srvy_l2	2022-11-04T18:4...	2020-02-2...	fgm	l2	mms_mms1	2020-04-03T00:...	2020-02-28...	3.6.3

# dataSummary DynamoDB

Summary of data holdings in fileRegistry DynamoDB; provides associations of missions with their data products, and enables users to find a data product string based on meta data

- Primary composite key: mission (Hash) + dataset (Sort)
- Secondary index: mission\_dataset
  - Query a mission and an instrument (e.g., mms\_mms1 + fgm) which will reveal all datasets associated with that combination
- Information includes:
  - Start and end data of the dataset
    - **Does not account for data gaps nor provide temporal resolution**
  - When the dataset was last updated
  - Variables contained within the dataset files (not guaranteed to be in every file and not applicable for some types of files)

dataSummary

Autopreview

View table details

▼ Scan/Query items

Scan/Query a table or index

Scan

Query

mission\_instr ▲

mission (Partition key)

Table

Enter partition key value

dataSummary

instrument (Sort key)

Index

Equal to ▼

Enter sort key value

mission\_instr ✓

9

► Filters

Run

Reset

mission	dataset	dataset_end	dataset_start	dataset_update	instrument	variables
mms_mms1	mms_mms1_fgm_srvy_l2	2020-04-11T...	2020-01-01T00:...	2022-11-04T18:4...	fgm	{"Epoch","Epoch_state","label_b_bcs","label_b_dmpa","label_b_gse","label_b_gsm","label_r_gse","label_r_gsm"}



# Reference Materials

DynamoDB primary keys and secondary indices

# DynamoDB Primary Keys

- **Primary key:** unique identifier for each item in a table, must be provided on insertion of item in table
  - *Using a table's primary key is the most efficient way to retrieve items while avoiding costly scans → primary key should be the dominant data search pattern*
  - **Simple primary key:** simple key-value storage (partition), e.g. user ID
    - Each item has a unique key
    - Example: The full CDAWEB filepath as a primary key  
“mms/mms1/fgm/srvy/l2/2015/09/mms1\_fgm\_srvy\_l2\_20150901\_v4.18.0.cdf”
  - **Composite primary key:** Partition + sort key, e.g. user ID + order date
    - Each item has a unique combination of partition + sort key, but multiple items may have the same partition key
    - Example: “mms1\_fgm\_srvy\_l2” + “20150901T000000”
    - Note: ISO-8601 time stamps are sortable in AWS DynamoDB
    - Enables more sophisticated querying of the table:
      - Grab all data within a partition
      - Grab all data in a partition covering the range of the sort key

# DynamoDB Secondary Indices

- **Secondary Index:** alternative identifier for items in the table which allows different querying, but do not have a uniqueness requirement
  - **Use secondary keys to avoid filtering data after doing a query**
  - **Local secondary index:** same partition key but has a different sort key, can only be used with a composite primary key
    - Example: ISO-8601 timestamp for the date the data was loaded, updated; file version number
    - ***Must be specified on table creation***
  - **Global secondary index:** defines an entirely different index for items across the entire database, could be a simple primary key, or a composite primary key
    - Primary key example: use the full CDAWEB filepath as a global secondary key → can get item by knowing the CDAWEB filepath and have a simple key structure; define MMS hash, MMS1 hash, FGM hash, instrument type hash (magnetometer, energetic particle, etc)
    - Secondary key example: Define “MMS” partition + “mms1\_fgm\_srvy\_I2\_20150901” (not recommended)
    - ***Can be added after a table has already been created***
- Not required for each item (primary keys are required); can create a sparse index where secondary keys are written for some items but not all
- Projected attributes: data to return from the entry when the table queries via the secondary index (cost sensitive)
  - Keys only: Will return only the keys for the index and the partition and sort keys for the table
  - All: provides the full table entry (expensive)
  - Include: Pick which attributes from the item to return
- **Secondary Index Limits:**
  - **20 global secondary indices and 5 local secondary indices can be created per table**
  - **10GB data stored per hash (since we are storing pointers to S3 data this shouldn't be an issue)**



JOHNS HOPKINS  
APPLIED PHYSICS LABORATORY