having moved the file, now editing the name
sandy_freelance authored just now

94b38dc4

SCR-registry-spec-0.3.md 23.87 KiB

Version 0.3.2 | HelioCloud |

1 The Shared Cloud Registry specification for HelioCloud

The Shared Cloud Registry (SCR) specification can be used for sharing datasets across cloud frameworks.

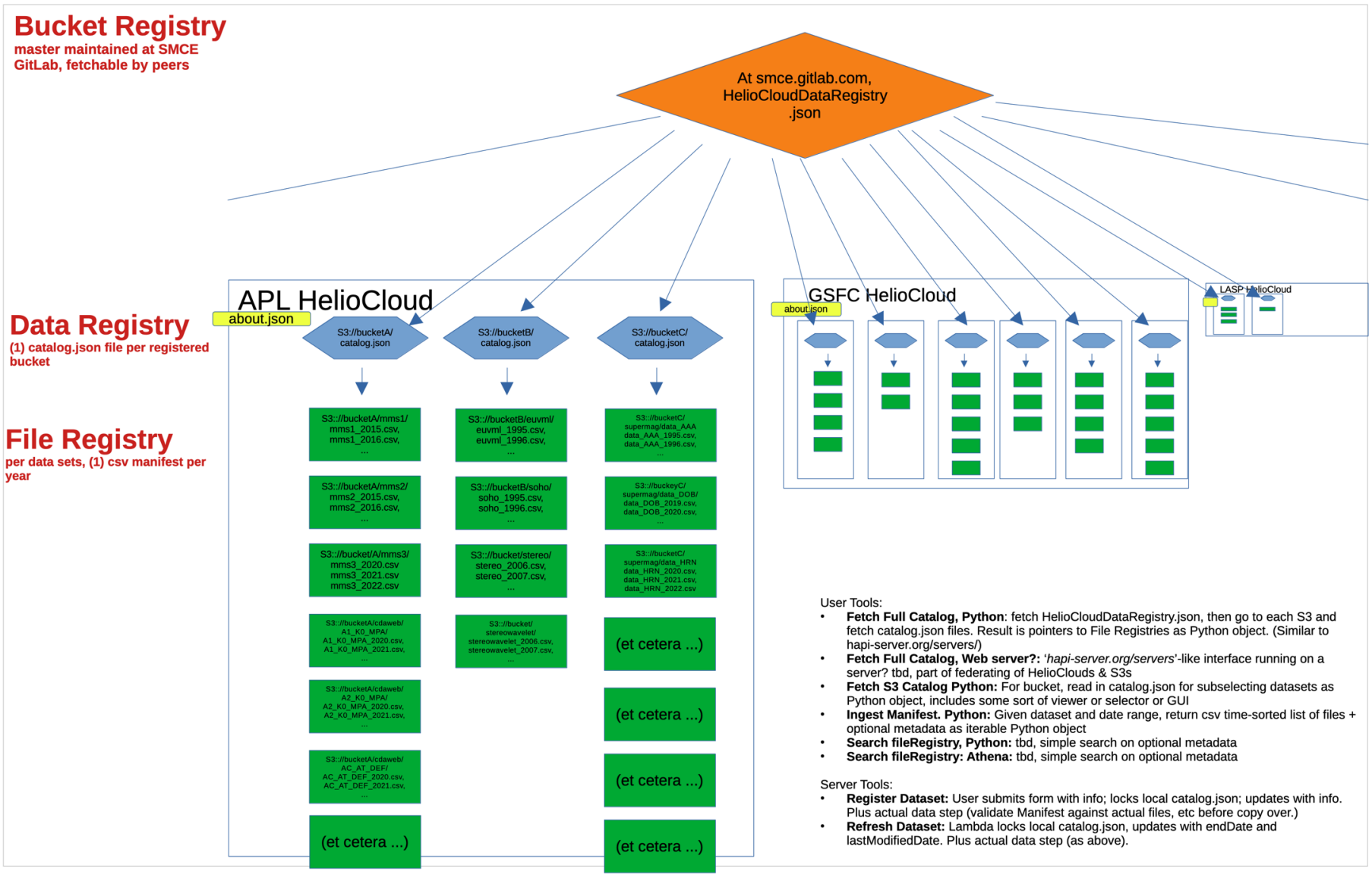
For HelioCloud, this specification creates a global data registry ('HelioDataRegistry'), maintained at the HDRL GitLab repository, then defines the file registry ('fileRegistry') for each dataset, that resides in the S3 (or equivalent) bucket alongside the dataset.

That 'HelioDataRegistry.json' is a JSON file consisting of **name** and **endpoint** and lists buckets as endpoints, not datasets. Tools can visit each **endpoint** to get the **catalog.json** listing datasets available in that bucket. The 'fileRegistry' consists of the required index to the actual files, and an optional dataset summary file. The fileRegistry itself is a set of ***_YYYY.csv*** (or csp-zip or parquet) index files, one per year. The optional summary file is named **.json** file and provides additional potentially searchable metadata.

1.1 Flow

Findability is through the 'HelioDataRegistry' JSON index of S3 buckets, which leads to the **catalog.json** for each S3 bucket that indicates available datasets, which points to the individual dataset file registry index files **_YYYY.csv** and its optional associated **.info** metadata auxillary file.

(Diagram here) github.com/whatever/HelioDataRegistry.json -> s3://aplcloud.com/mybucket/catalog.json -> individual dataset fileregistries(.csv)



Datasets are findable by going to 'HelioDataRegistry.json' to get endpoints, then visiting each S3 endpoint to get the catalog listing of datasets available.

Accessing the actual data involves accessing each dataset's file registry index files. The fileRegistry consists of CSV files, one per year. Therefore, a user has the choice to:

- (a) download the CSV file
- (b) use our provided Python API to fetch a subset of the CSV file, by date range
- (c) use AWS Athena to run queries on the CSV file

1.2 Uploading Data

Data providers must provide both the Data. the Dataset Description (in .json as per the catalog.json spec) and a Manifest (in the .csv format we specify).

We will generate an S3 Inventory of the contents and check against their Manifest, and send them the 'diff' file. It is their responsibility to provide us with a corrected Manifest. This iterates until their Manifest matches the actual Data provided.

1.3 Refreshing Data

Periodic updates to datasets go into an upload bucket. The process is generally the same as for Uploading Data: provide Data and Manifest. Once the Manifest matches the Data, we move the data from the upload bucket to the proper location and append the new Manifest to the existing Manifests.

1.4 Scope

We will produce Python tools that fit this API; a web form or similar tool for users to register datasets; automation to move datasets from an upload bucket to the data buckets; scripts to generate the AWS Inventory and compare against a user-provided Manifest; a sample use of Athena against the fileregistry for queries.

This meets requirements 1, 4, 5, 6, 7 and 11 from the user stories/requirements document,, and opens up 9 (current scope is data be findable, whereas item 9 adds searchability).

2. Global data registry 'HelioDataRegistry.json'

The global data registry is an unsorted json list of names and bucket endpoints. This is for buckets, not datasets (a bucket can hold multiple datasets). Queries to each endpoint can fetch the list off datasets and more information from the **endpoint**/catalog.json file.

The definitive listing of available dataset is kept in the **name for GSFC HelioCloud registry**. Providers who wish to make data visibile to the public do a one-time registration of their S3 bucket (or equivalent) to the main registry The global data registry can be accessed directly or mirrored by anyone.

Once an S3 bucket (or equivalent) is registered in the HelioDataRegistry.json file, it does not have to be updated when new datasets are added, only when new buckets are made public.

For this global registry, only S3 buckets, not subbuckets, are allowed. For example, 's3://helio-public/' is allowed, but 's3://helio-public/MMS/' is not.

2.1 Global Data Registry specification

The registry is a JSON file containing the version of this specification, the date it was last modified, and the registry list of datasets. Each dataset has two items: **endpoint** and **name**. **endpoint**s must be unique; names do not enforce uniqueness.

- **endpoint** - An accessble S3 (or equivalent) bucket link
- **name** - A descriptive name for the dataset.
- **provider** - default 'aws', indicates which cloud provider. Included for future federation.
- **region** - The AWS or equivalent region the bucket resides in

The **name** should be brief and one-line; length is not enforced by the specification but may be truncated when registration to keep things readable.

2.2 Sample global data registry

Here is a sample 'HelioDataRegistry.json' for three buckets at two sites.

```
{
    "version": "0.3",
    "modificationDate": "2022-01-01T00:00.00Z",
    "registry": [
        {
            "endpoint": "s3://gov-nasa-hdrl-data1/",
            "name": "GSFC HelioCloud Set 1",
            "provider": "aws",
            "region": "us-east-1"
        },
        {
            "endpoint": "s3://gov-nasa-hdrl-data2/",
            "name": "GSFC HelioCloud Set 2",
            "provider": "aws",
            "region": "us-east-1"
        },
        {
            "endpoint": "s3://edu-apl-helio-public/",
            "name": "APL HelioCLoud",
            "provider": "aws",
            "region": "us-west-1"
        }
    ]
}
```

3. Catalog of File Registries (per bucket catalog.json)

The catalog.json file has an entry for each dataset stored within the given S3 bucket. **endpoint** and **name** are identical to the item in the global data registry.

Globally the catalog.json describes the endpoint with the following items. Note that **endpoint** and **name** are included for clarity sake (being duplicates of the main registry) as JSON does not allow for comments, and should be the same as was provided to the global registry.

- **endpoint** - same as was provided to GlobalDataRegistry.json, an accessble S3 (or equivalent) bucket link
- **name** - same as was provided to GlobalDataRegistry.json, a descriptive name for the dataset.
- **region** - same as was provided to GlobalDataRegistry.json, which AWS region
- **egress** - one of 'no-egress', 'user-pays', 'egress-allowed', or 'none'
- **status** - A return code, typically "1200/OK". Site owners can temporarily set this to other values
- **contact** - Who to contact for issues with this bucket, e.g. "Dr. Contact, dr_contact@example.com"
- **description** - Optional description of this collection
- **citation** - Optional how to cite, preferably a DOI for the server
- **comment** - A catch-all comment field for data provider and developer use. It should not contain information required to parse the data items.

For each dataset, the catalog entry requires:

- **id** a unique ID for the dataset that follows the ID naming requirements
- **index** a fully qualified pointer to the object directory containing both the dataset and the required fileRegistry. It MUST start with s3:// or "https://" (or equivalent) end in a terminating '/'.
- **start**: string, Restricted ISO 8601 date/time of first record of data in the entire dataset OR the word 'static' for items such as model shapes that lack a time field.
- **stop**: string, Restricted ISO 8601 date/time of end of the last record of data in the entire dataset OR the word 'static' for items such as model shapes that lack a time field.
- **modification**: string, Restricted ISO 8601 date/time of last time this dataset was updated
- **title** a short descriptive title sufficient to identify the dataset and its utility to users
- **indextype** Defines what format the actual fileRegistry is, one of 'csv', 'csv-zip' or 'parquet'
- **filetype** the file format of the actual data. Must be from the prescribed list of files.
- **description** optional description for dataset".
- **resource** optional identifier e.g. SPASE ID, dataset description URL, DOI, json link of model parameters, or similar ancillary information".
- **creation** optional ISO 8601 date/time of the dataset creation".
- **expiration** optional ISO 8601 date/time after which the dataset will be expired, migrated, or not maintained".
- **verified** optional ISO 8601 date/time for when the dataset was last tested by verifier programs".
- **citation** optional how to cite this dataset, DOI or similar".
- **contact** optional contact name".
- **about** optional website URL for info, team, etc.
- **multiyear** optional True/False field (default: False) for use when dataitems span multiple years (see 3.2 below)

For file formats, there is a prescibed list. As new file formats are introduced, we will update this specification to give a single unique identifier for it. The reason for the prescribed list is to avoid ambiguity or the need for users to parse (for example, avoiding figuring out '.fts', 'fits', '.FTS', etc) Repositories with multiple files types can specify them as a

comma-separated list with no spaces, e.g. 'fits,csv' for a dataset that contains both images and event lists. Currently defined types are 'fits,csv,cdf,netcdf3,netcdf4,hdf5,datamap,txt,binary,other'.

The catalog.json file has to be updated when new data is added to a dataset, by updating the **stop** item. Also, the catalog.json file is updated when a new dataset is put into that S3 bucket.

Note, currently this specification is defined around "s3://" architecture with some support for "https://" endpoints; future versions may support other protocols.

3.1 ID naming requirements

The **id** field can only contain alphanumeric characters, dashes, or underscores. No spaces or other characters are allowed.

The **id** field will match the fileRegistry files but does not have to match the sub-bucket names. The fileRegistry includes the **id**_YYYY.csv file indices and the optional **id**.json metadata file.

3.2 Data items spanning multiple years

Some data or model outputs span multiple years. An output product that covers 3 years, for example (2011-2013) would be index in "id"_2011.csv (based on the start date of the data). A time-based search that is looking for 2012 coverage would therefore not be able to find it, as no "id"_2012.csv file exists or is needed. To accommodate these edge cases, the **multiyear** field should be set to True, which will allow subsequent search layers to be aware that long-duration files exist in this dataset.

This field should not be set for the typical case where a datafile extends slightly into the next year, but only when a datafile exists to provide data for a calendar year and that datafile would not be findable based purely on its given start date.

3.2 Status codes

The default status code for a catalog.json item is "1200/OK" indicating the data is available. Status codes are informative and do not enforce any limits. They exist to communicate to users and client programs if a dataset is temporarily down or has other constraints. Other status codes defined so far include:

- "code": "1200", "message": "OK": system is up and running
- "code": "1400", "message": "temporarily unavailable": providers can set this if they temporarily are doing maintenance or need to stop access due to costs

3.3 Example

Here is an example catalog, for which only the first item has decided to fill out the optional 'ownership' block. If this was the GSFC catalog.json, it would reside at s3://gov-nasa-helio-public/catalog.json.

```
{
    "version": "0.3",
    "endpoint": "s3://gov-nasa-helio-public/",
    "name": "GSFC HelioCloud",
    "region": "us-east-1",
    "egress": "no-egress",
    "contact": "Dr. Contact, dr_contact@example.com",
    "description": "Optional description of this collection",
    "citation": "Optional how to cite, preferably a DOI for the server",
    "catalog":[
        {
            "id": "euvml",
            "index": "s3://gov-nasa-helio-public/euvml/",
            "title": "EUV-ML dataset",
            "start": "1995-01-01T00:00.00Z",
            "stop": "2022-01-01T00:00.00Z",
            "modification": "2022-01-01T00:00.00Z",
            "indextype": "csv",
            "filetype": "fits",
            "description": "Optional description for dataset",
            "resource": "optional SPASE ID, DOI, URL, or json modelset",
```

```
            "creation": "optional ISO 8601 date/time of the dataset creation",
            "citation": "optional how to cite this dataset, DOI or similar",
            "contact": "optional contact name",
            "about": "optional website URL for info, team, etc"
        },
        {
            "id": "mms_hmi",
            "index": "s3://gov-nasa-helio-public/mms/hmi/",
            "title": "MMS HMI data"
            "start": "2015-01-01T00:00.00Z",
            "stop": "2022-01-01T00:00.00Z",
            "modification": "2022-01-01T00:00.00Z",
            "indextype": "csv-zip",
            "filetype": "cdf"
        },
        {
            "id": "mms_feeps",
            "index": "s3://gov-nasa-helio-public/mms/feeps/",
            "title": "MMS FEEPS data"
            "start": "2015-01-01T00:00.00Z",
            "stop": "2022-01-01T00:00.00Z",
            "modification": "2022-01-01T00:00.00Z",
            "indextype": "csv-zip",
            "filetype": "cdf"
        },
        {
            "id": "fluxrope",
            "index": "s3://heliotest/models/",
            "title": "Instatiation of 3D fluxropes"
            "start": "static",
            "stop": "static",
            "modification": "2022-01-01T00:00.00Z",
            "indextype": "csv-zip",
            "filetype": "cdf"
        }
    ],
    "status": {
        "code": 1200,
        "message": "OK request successful"
    }
}
```


3.4 Indexes should reside in same bucket as data

Index CSV files must be in the same bucket as the data, but do not have to be in the same sub-bucket or directory as their data. The catalog points to the index files, and the index files use absolute paths to point to the data items.

This also enables design a collection of datasets, wherein the data in the actual file registry _.csv files can span sub-buckets. The use of absolute file paths is mandated.

```
Example data itself is in:
  s3://example/mms1/feeps/
  s3://example/mms2/feeps/
  s3://example/mms3/feeps/
  s3://example/mms4/feeps/
```

```
Case 1: Matching
    file registry locations:
         s3://example/mms1/feeps/mms1_feeps.CSV
         s3://example/mms2/feeps/mms2_feeps.CSV
         s3://example/mms3/feeps/mms3_feeps.CSV
         s3://example/mms4/feeps/mms4_feeps.CSV
```

```
Case 2: Not Matching
    file registry locations:
         s3://example/mms_all/feeps/mms_feeps.CSV (contents point to 4 subbuckets)
```

The first case matches the idea of a 'dataset' and MUST be provided for any provided dataset. The second matches the idea of a 'collection' and is supported but not required (i.e. additional extra fileregistry endpoints do not have to match the underlying data structure).

3.5 Concerns

Concerns were voiced about clashes if multiple people attempt to edit the catalog.json for a bucket at the same time. Our default HelioCloud will maintain the catalog.json contents in a serverless DynamoDB (which will handle transaction collisions and provide rollback), which then outputs the 'catalog.json' file that users and data requests use.

4 File Registry

The file registry consists of one index for each year of the dataset in either csv, zipped csv, or parquet format. The file must be in time sequence and the first three items must be the **start**, **datakey**, and **filesize** fields in that order.

The index fileRegistry is a set of CSV or Parquet files named "index"/"id"_YYYY.csv, "index"/"id"_YYYY.csv.zip or "index"/"id"_YYYY.parquet. For the case of static non-time sequence outputs, the index fileRegistry are named "index"/"id"_static.csv (or .csv.zip or .parquet).

4.1 Required Items

- **start**: string, Restricted ISO 8601 date/time of start for that data OR the word 'static' for items such as model shapes that lack a time field
- **datakey**: string, full filename or S3 object identifier sufficient to actually obtain the file
- **filesize**: integer, file size in bytes

4.2 Optional Items

- **stop**: string, Restricted ISO 8601 date/time of end for that data.
- **checksum**: checksum for that file. If given, **checksum_algorithm** must also be listed
- **checksum_algorithm**: checksum algorithm used if checksums are generated. Examples include SHA, others.

4.3 Optional Parameters

The default expected search capability is [time range) (start time within desired range).

Anything past **filesize** is fully optional; the minimal API expects only a start time, datakey aka filehandle, and file size IN THAT ORDER in the actual file index. It is up to individual client programs to do anything past that.

Any metadata included in this per-file index must be defined in the .info json file to allow parseability.

The csv or zipped csv files may or may not include a one-line header that is prefaced by "#". It is the responsibility of client programs to determine if there is a skippable header or not.

Any optional parameters from the above list or the info JSON must be in the same order specified in the JSON.

For 'static' items, since the 'start' field will be the same constant value 'static' for all items, optional parameters to distinguish the items are suggested.

4.4 Accessing

Users have 3 options with the fileRegistry CSV file:

- download it directly and parse yourself,
- use our Python API (provided) to extract a subset of filehandles from CSV,
- use AWS Athena for queries

4.5 Justification for Yearly CSV/Parquet Files

Unlike a database, yearly index files are both fetchable and parseable. They have a lower cost profile than the equivalent database, reasonably fast access, and allow for client and search programs independent of a specific database implementation.

Using yearly files rather than a single file or a database is to maintain an inexpensive, stateless, easily updated file index. In AWS, the "S3 Inventory" command can generate a list of filenames and datakeys, or filenames and datakeys since the last time inventory was run. Being able to add to the file registry index files incrementally is easier served if they are chunked into yearly files.

In addition, many use cases for long time baseline datasets will not need to access the entire multi-decadal span of the data, so parsing into years reduces the downloads needed to obtain the indices.

The use of CSV or Parquet also enables AWS Athena searches within the index with little overhead, so long as optional metadata is provided.

4.6 Example File Registry

Here is a short minimal CSV example index file.

```
# start, datakey, filesize
'2010-05-08T12:05:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120530_n4euA.fts','246000'
'2010-05-08T12:06:15.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120615_n4euA.fts','246000'
'2010-05-08T12:10:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_121030_n4euA.fts','246000'
```

Here is an example with additional metadata and a CSV header as well.

```
# start, datakey, filesize, wavelength, carr_lon, carr_lat
'2010-05-08T12:05:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120530_n4euA.fts','246000','195'
'2010-05-08T12:06:15.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120615_n4euA.fts','246000','195'
'2010-05-08T12:10:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_121030_n4euA.fts','246000','195'
```

Here is an example with additional metadata and a CSV header as the EUV-ML project would like. Items in CAPS are directly from FITS keywords.

```
# start, datakey, filesize, spacecraft, instrument, WAVELNTH, CRLT_OBS, CRLN_OBS, CRPIX1, CRPIX2, RSUN, quality, g
'2010-05-08T12:05:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120530_n4euA.fts','246000','A','
'2010-05-08T12:06:15.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_120615_n4euA.fts','246000','A','
'2010-05-08T12:10:30.000Z','s3://edu-apl-helio-public/euvml/stereo/a/195/20100508_121030_n4euA.fts','246000','A','
```

5 Time Specification and ISO 8601

Time values are always strings, and the SCR Time format (taken from the HAPI Time format) is a subset of the ISO 8601 standard. The restriction on the ISO 8601 standard is that time must be represented as

```
yyyy-mm-ddThh:mm:ss.sssZ
```

and the trailing Z is required. Strings with less precision are allowed as per ISO 8601. Any date or time elements missing from the string are assumed to take on their smallest possible value. For example, the string 2017-01-15T23:00:00.000Z could be given in truncated form as 2017-01-15T23:00Z. A dataset must use only one format and length within that given dataset. The times values must not have any local time zone offset, and they must indicate this by including the trailing Z.

6 Info Metadata

Each dataset may also include an optional info json file that gives the time range, date last modified, ownership information, and optional additional metadata for that dataset. The files are by default searchable and selectable by time window. Additional search capability is not within scope of the file registry per se, but data providers can

indicate metadata for adding a search layer.

If an .info file exists and lists additional parameters, the resulting file Registry index files must contain those parameters in the same order as expressed in this json file.

6.1 Optional Items

- **parameters**: optional list of searchable parameters available in the actual file registry index files file. (* = available from S3 Inventory)

6.2 Example

Here is an example for a sample optional Info json file. This is used to indicate additional searchable metadata that exists within the file Registry index files, and enables higher searchability in datasets.

```
{
    "version": "0.3",
    "parameters": [
        {"name": "spacecraft", "type": "string"},
        {"name": "wavelength", "type": "int", "units": "Angstroms"},
        {"name": "crlt", "type": "double", "units": "degrees", "desc": "Carrington latitude"},
        {"name": "crln", "type": "double", "units": "degrees", "desc": "Carrington longitude"},
        {"name": "rsun", "type": "double", "units": "pixels", "desc": "Size of sun in pizels"},
        {"name": "crpix1", "type": "integer", "units": "pixels", "desc": "x coord of sun center"},
        {"name": "crpix2", "type": "integer", "units": "pixels", "desc": "x coord of sun center"},
        {"name": "quality", "type": "integer", "desc": "data quality and level of interpolation"}
    ]
}
```

7.0 Changes from 0.2 to 0.3

Changed

```
startDate -> start
stopDate -> stop
modificationDate -> modification
egressPolicy -> egress
contactURL -> URI
aboutURL -> about
indexFormat -> indextype
fileFormat -> filetype

key -> datakey
loc -> index

Removed 'resourceURL' as redundant

Added 'expiration' and 'verified' fields
Added 'comment' field
Added 'static' as a time option for start/stop for use with items that are not time-tagged, such as model outputs.

Added multiyear flag to handle (mostly model) case where a single datafile will span multiple years.  Clarificatio
```