Pacific Northwest
NATIONAL LABORATORY

PNNL-XXXXX

ME Data Pipeline Standards: Version 1.0

March 2021

Carina Lansing
Maxwell Levin
Chitra Sivaraman

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <https://www.ntis.gov/about>
Online ordering: http://www.ntis.gov

PNNL-XXXXX

ME Data Pipeline Standards: Version 1.0

March 2021

Carina Lansing
Maxwell Levin
Chitra Sivaraman

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Acknowledgments

This material is based upon work conducted in support of the Water Power Technologies Office within the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy.

Acronyms, Abbreviations, and Glossary

ACT	Atmospheric data Community Toolkit [https://github.com/ARM-DOE/ACT]
API	Application Programming Interface
ARM	Atmospheric Radiation Measurement
CF	Climate/Forecast
Data Lake	A data lake is a centralized repository that stores an organization or project's data at any scale. Data lakes enable analysts and scientists to easily query and analyze data across multiple sources.
Dataset	A combination of variables, dimensions, and metadata describing data over some period of time. Datasets are programming objects that can be loaded from one or more files or data lake queries.
Datastream	Set of all data collected over all time at a specific location, instrument, and data level.
Developer	Person responsible for data pipeline software development
DOI	Digital Object Identifiers
Ingest	A data product created by processing raw data into standard format with quality control checks applied.
Instrument	A single piece of hardware or group of sensors hardware that records one or more measurements.
ME	Marine Energy
Mentor	Person responsible for instrument installation and general operations.
Metadata	Information describing data such as how the data were collected, where the data were collected, any algorithms used, and quality of the data. Metadata are used to facilitate search and discovery, reproduce results, and ascertain confidence in the data.
MHKDR	Marine and Hydrokinetic Data Repository
MHKiT	Marine Hydrokinetic Toolkit
MRE	Marine Renewable Energy (This acronym has been renamed to Marine Energy)
NaN	not a number indicator
NetCDF	Network Common Data Form
QC	Quality Control
RAW	Primary data collected by an instrument that has not been processed yet.
Site	Geographical location where a set of measurements are being conducted.

URL	Universal Resource Locator. The address of a web page.
UTC	Coordinated Universal Time
VAP	Value-Added Product; a higher-order data product that includes derived quantities not measured directly or routinely. VAPs provide an important translation between the instrumental measurements and the geophysical quantities needed for scientific analysis.

Contents

Figures

Tables

1.0 Introduction

Working with standardized data provides many benefits that can accelerate analyses of instrument and model-based data and increase their reliability. Some of these benefits include:

- Adherence to International Electrotechnical Commission Standards
- Improved data quality and traceability
- Improved data compatibility, reuse, and interoperability
- Reduced redundancy, time, and effort spent on reconciliation and statistical analysis
- Consistent analysis of multiyear data
- Easier integration with standard analysis libraries (e.g., MHKiT [Marine and Hydrokinetic Toolkit])
- Reduced costs

In order to facilitate the standardization of data produced by the marine energy community, the U.S. Department of Energy's Water Power Technologies Office is currently funding a data pipeline framework that can be easily used to convert data into a standardized, verified, interoperable format. The goal is for projects to be able to quickly produce a scalable archive of standardized, reliable data that can be easily accessed, integrated, and analyzed via MHKiT [1] libraries, incorporated into machine learning algorithms for long-term prediction capabilities, and transferred to the Marine and Hydrokinetic Data Repository (MHKDR) for final archival and publication.

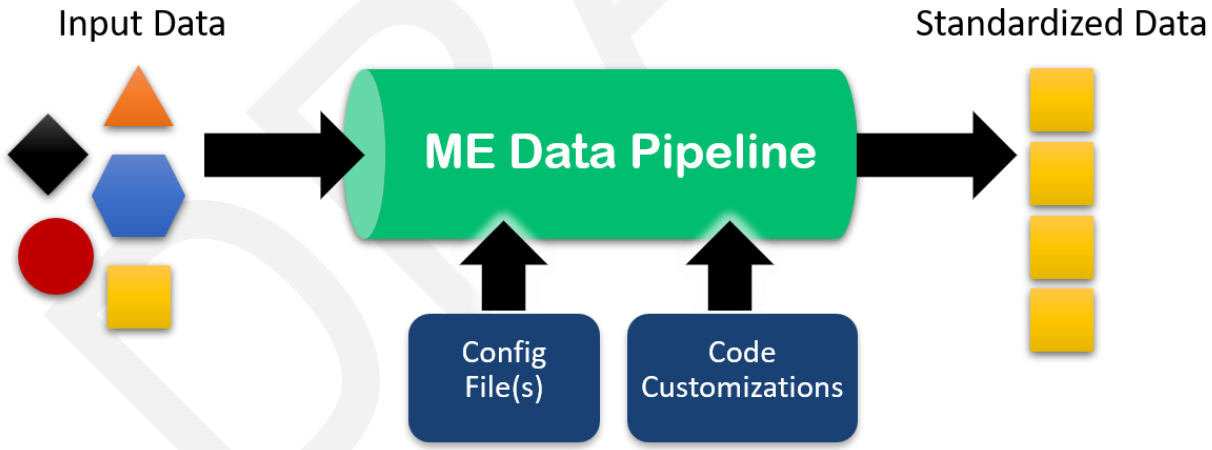

Figure 1. Marine Energy (ME) Data Pipeline

This document presents an **initial** baseline set of data standards that will govern all data produced by ME data pipelines. It is intended for anyone writing ME data pipelines or using data produced by ME data pipelines. This document specifically focuses on the data standards (i.e., the output of the data pipelines). For more information on the pipeline framework itself, please see the documentation at https://tsdat.readthedocs.io/.

This document draws upon established Climate/Forecast (CF) data standards that have been well vetted for the Atmospheric Radiation Measurement (ARM) program over the past 20 years

[2,3]. Multidimensional data formats, common metadata, geospatial information, file naming conventions, and quality control information are some of the scientific data topics covered by these standards. As these topics align with and directly apply to ME data, they have been carried over into this document. However, each topic has been reviewed against other published marine energy and ocean data standards [4,5,6,7,13] and tailored accordingly.

This is a dynamic document which is meant to continuously evolve as the findings from ongoing research and development lead to their refinement. Additional tailoring is expected as this document is reviewed by the greater ME community.

2.0 ME Data Pipelines

ME data pipelines will produce data that conform to the standards defined in this document. Specifically, the pipelines will make sure that:

1. Data are converted to standard engineering units
2. Data variables are named consistently
3. Data are annotated with enough metadata so that users will understand how the data values were produced
4. Data are corrected as needed with specific instrument/measurement calibrations
5. Data are passed through a set of customizable, automated quality control/quality assurance checks, and the results of the checks and any corrections made are stored with the data so they are not lost
6. New data variables can be specified and derived via scientific algorithms
7. Supplemental data plots can be produced and stored adjacent to the data to assist with manual review activities
8. Pipeline status can be monitored so that critical data errors can be addressed in near real time and resolved as quickly as possible
9. Data produced by the pipeline can be easily opened, filtered, combined, and analyzed via MHKiT Application Programming Interfaces (APIs) either using cloud resources such as a data lake or on a local filesystem

Pipeline developers will be able to define the specific variables, metadata, corrections, and quality tests used for each pipeline via a set of configuration files and a code template. Detailed information on configuring and running an ME pipeline is provided here: https://tsdat.readthedocs.io. **This document focuses on identifying the specific format of the processed data and baseline metadata that will be captured.**

3.0 Dataset Terminology

We refer to a **dataset** as an object for storing data (variables) such as temperature, voltage, wave height, etc. Each of these variables can be collected across one or more dimensions (such as time or height) and can be annotated with specific metadata describing how the variable was collected, its quality status, and any corrections that were applied. In addition, the entire dataset can be annotated with metadata describing the instruments involved, location, and specific data collection procedures. **The combination of variables, dimensions, and metadata represents a dataset**. Individual datasets represent the variables collected over some subset of time. The collection of all datasets produced for a given location, instrument, and data pipeline across all time is referred to as a **datastream**.

Datasets may be stored in specific file formats and/or they may be loaded into memory and accessed via an API. ME data pipelines will utilize well-known, established APIs and file formats for working with datasets. Specifically, ME data pipelines will utilize the XArray library [8] for working with datasets via Python code, and will utilize Network Common Data Form (NetCDF) [9] for storing dataset data to file. (Other file formats will also be supported, as described in the pipeline documentation at https://tsdat.readthedocs.io/.)

As XArray and CF standards both follow NetCDF data conventions, we will use the following NetCDF terminology to refer to the various constructs within the dataset:

Attributes: Metadata describing either the dataset/file (*Global Attributes*) or a specific variable (*Variable Attributes*)

Dimensions: The points at which data values are collected; also known as the shape of the data (e.g., “time,” “height,” “depth,” etc.)

Coordinates: The values/indexes for each dimension.

Variables: A labeled n-dimensional data array containing the values of a specific property (e.g., voltage) dimensioned along zero or more dimensions (e.g., time). Each variable can contain a number of self-describing metadata attributes.

For more information on XArray data structures, please consult the XArray documentation at [http://xarray.pydata.org/en/stable/data-structures.html].

4.0 General Dataset Requirements

This section describes the general dataset requirements (metadata attributes, dimensions, and variables) that should apply to any dataset produced by an ME data pipeline. This dataset format is built upon the well vetted netCDF format used by the climate community. These standards are designed to make sure that data can be easily combined and analyzed across multiple datastreams and that the data are self-describing, with appropriate metadata included to explain how/where the data were captured, any algorithms/corrections that were applied, and its quality. **In addition to the general requirements defined here, users should also check the registered MHKDR content models to see if one of these applies to their data, as described in Section 5.**

4.1 Global Attributes

Global attributes are metadata applied at the dataset level which describe the data and how it was derived.

Attributes are defined by a key/value pair. The key, or name, should begin with a letter and be composed of letters, digits, and underscores. The value can be a single value or an array of values (*any global attributes mentioned in this document that are referred to as plural, we be assumed to be an array of values*). Recommended attributes may be omitted if the value is expected to be unknown.

This section describes the minimum set of global attributes that could be included in a dataset. These attributes are listed below in alphabetical order for ease of reference. However, it is recommended that the **title** and **description** attributes appear first when written to file to assist with manual browsing of the data. Any attributes which are auto-generated by the pipeline are indicated below with a bold “auto-generated” label.

Table 1. Required and Recommended Global Attributes

Attribute Name	Required?	Description
averaging_interval	no	If instrument data is not instantaneous and each time value represents the average over a specific time interval, records expected averaging interval (e.g., “5 minute”).
code_url	yes	Include the url to the specific tagged release of the repository used for this pipeline invocation. For example, https://github.com/clansing/twrmr/releases/tag/1.0.
collection_method	no	An optional string representing the collection method that was used for this data. The value for collection_method should be selected from the MHKDR taxonomy provided in Appendix A. Example: “Field Data.”
Conventions	yes	The version of the standards document this data conforms to. Example: “ME Data Pipeline Standards v1.0.” **auto-generated.**
data_level	yes	The processing level of the data (e.g., a1, b1, etc.). See Section 7 for more information.

Attribute Name	Required?	Description
dataset_name	yes	A string consisting of any letters, digits, "-", or "_" that can be used to uniquely identify the data product being produced. To prevent confusion with the temporal resolution of the instrument, the dataset name must not end with a number.
datastream_name	yes	This attribute will be automatically recorded via the ME data pipeline engine and is constructed like: (location_id).(dataset_name)(qualifier)(temporal).(data_level) **auto-generated.**
description	yes	User-friendly description of the dataset. It should provide enough context about the data that new users are able to quickly understand what the data is measuring and how it can be used. Example: "ARM best estimate hourly averaged Quality Control (QC) controlled product, derived from ARM observational Value-Added Product data: ARSCL, MWRRET, QCRAD, TSI, and satellite; see input_files for the names of original files used in calculation of this product."
doi	no	Digital Object Identifiers (DOIs) are not required. However, if one has been obtained for the data, it may be included here. Example: "10.5439/1039926" Note this is a character string.
history	yes	A record of who ran the pipeline to produce this data, when it ran, and where it ran. This attribute will be automatically recorded via the ME data pipeline engine. **auto-generated**
input_files	yes	Multi-valued attribute that lists all the input files used to produce this dataset. This attribute will be automatically recorded via the ME data pipeline engine. If multiple files in a continuous time block are used, the first and last file may be specified, connected with a hyphen. Example: "sgpqcrad1longC1.c2.20140417.000000 - sgpqcrad1longC1.c2.20140719.000000" **auto-generated**
institution	no	Specifies the institution who produced the data.
instrument_description	no	Records a description of instrument(s) used to collect data.
instrument_manufacturer	no	Records the manufacturer of instrument(s) used to collect data.
instrument_name	no	Records name of instrument(s) used to collect data.
last_calibration_date	no	Records the date(s) of the last time the instrument(s) was calibrated.
location_description	no	A description of the location where the data were obtained from.
location_id	yes	A label or acronym for the location where the data were obtained from.
qualifier	no	An optional qualifier that distinguishes these data from other data sets produced by the same instrument or downstream products. The qualifier must not end with a number.
references	no	Published or web-based references that describe the methods algorithms, or third-party libraries used to process the data. For example: https://github.com/MHKiT-Software/MHKiT-Python

Attribute Name	Required?	Description
sampling_interval	no	Records expected sampling interval of the instrument (for example, “400 us” or “60 Hz”).
sensor_height	no	The height of the instrument above-ground or water-surface level. A negative value represents a measurement below ground level. Example: `sensor_height = "10.5 m AGL."` *This attribute may be overridden at the variable level if the variable is measured at a different height than the instrument.
serial_number	no	Records serial number of instrument(s) used to collect data.
technology	no	An optional string representing the marine technology that this data references. The value for technology should be selected from the MHKDR taxonomy provided in Appendix A. Example: “Axial Flow Turbine”
temporal	no	An optional description of the data’s temporal resolution (e.g., 30m, 1h, 200ms, 14d, 60hz). All temporal resolution descriptors require a units identifier. Accepted abbreviations include ns=nanosecond, us=microsecond, ms=millisecond, s=second, m=minute, h=hour, Hz=hertz, d=day, mo=month, yr=year.
title	yes	A succinct English language description of what is in the data set. The value would be similar to a publication title. Example: “Marine Energy Offshore Buoy Surface Temp (MEBESFCTEMP)”
topic	no	An optional string representing the marine energy topic that this data pertains to. The value for topic should be selected from the MHKDR taxonomy provided in Appendix A. Example: "Technology"

As described above, this section provides only a minimal set of provenance metadata that should be captured. Users are free to add any additional global attributes required to adequately describe their data.

4.2 Dimensions

Dimensions are used to specify the shape of one or more of the variables contained in a dataset. Dimensions also provide a way to index the variables contained in the dataset. A dimension may be used to represent a real physical dimension (e.g., time, depth, height) or a more abstract quantity (e.g., bin, instrument number, model-run ID, etc.).

Every dimension has both a name and a size. The name should begin with a letter and be composed of letters, digits, and underscores. It is preferred that dimension names use all lowercase letters when possible. The size is an arbitrary positive integer, or it can have the special size **unlimited**. Such a dimension is called an unlimited dimension or a record dimension. A variable with an unlimited dimension can grow to any length along that dimension. A dataset need not have any unlimited dimensions, but time is the typical unlimited dimension. This is so that the number of time points does not have to be known beforehand, and any number of time points can be included in a file. We recommend that the number of dimensions used in a single file be as few as possible.

4.3 Variables

A variable represents a multidimensional array of values of the same type. A dataset can contain multiple variables, each storing the values for a specific measurement or physical quantity. This section describes how to describe dataset variables and the standard variables and variable metadata that apply to datasets produced ME data pipelines.

4.3.1 Variable Format

A variable is specified by its name, a list of dimensions (dims) which describe its shape, an n-dimensional array of its data, and a set of additional metadata attributes that describe it further. The rules for specifying a variable's name, dims, and metadata attributes are described below:

name Variable names should begin with a letter and be composed of letters, digits, and underscores. For more guidance on choosing a variable name, see the ARM Data Standards, Section 6.4 [3].

dims Each variable must include a vector field called dims that list the dimension names for that variable (e.g., ['time', 'depth']). A variable with an empty dims vector has no dimensions and consists of a single scalar value.

data Each variable stores measured or derived data in an N-dimensional array whose shape is given by the number and length of each of its dimensions. The number of values stored in a non-scalar variable is the product of the lengths of all its dimensions.

attrs Described in Section 4.3.2 – Variable Attributes and Section 4.5.1 – QC Variable Attributes

4.3.2 Variable Attributes

Variable attributes provide additional metadata about the variable and how it was processed. This section describes standard variable attributes that apply to all variables. For ME data pipelines, variable attributes may be specified in the accompanying configuration files. A collection of required and recognized optional variable attributes is listed in Table 2.

Table 2. Required and Recommended Variable Attributes

Attribute Name	Required?	Description
_FillValue	no	Value used to initialize the variable's data and to indicate that the data is missing. Defaults to -9999. Not allowed for coordinate variables. Should be a value that could not reasonably be mistaken for a physical value or data point.
ancillary_variables	no	Array of variables that depend on the values from this variable. This is primarily used to indicate if a variable has an ancillary qc variable. *QC ancillary variables will be automatically recorded via the ME data pipeline engine.*

Attribute Name	Required?	Description
cell_methods	no	If the data has been transformed onto a different time grid, this attribute indicates the statistical method that was used to transform the data. This attribute is well defined by the CF convention and is extendable to describing multidimensional data sets. Additional description can be found at cfconventions.org.
comment	no	A user-friendly description of what the variable represents, how it was measured or derived, or any other relevant information that increases the ability of users to understand and use this data.
corrections_applied	no	Array of strings indicating what corrections, if any, have been applied to the data.
fail_delta	no	Largest difference between any two consecutive values in the data above which should be treated with heavy skepticism. If applying QC tests, values exceeding this delta should be given a “Bad” assessment.
fail_range	no	Two-element array of [min, max] outside of which the data should be treated with heavy skepticism. If applying QC tests, values outside of this range should be given a “Bad” assessment.
long_name	yes	Brief label to describe what the variable represents. Used by several netCDF libraries to generate axes labels in plots.
source	no	Optional machine-readable attribute to describe the datastream and variable that this variable was retrieved from or (optionally) the algorithm used to calculate the variable. The format for the source attribute if the variable is retrieved is: *datastream:variable_name*. If the variable is calculated, the source attribute is just the name of the algorithm, excluding any “:” characters.
standard_ name	yes/no	A string exactly matching a value in the CF or ME Standard Name table*. Used to provide a standardized way of identifying variables and measurements across heterogeneous datasets and domains. Not required if a suitable standard name for the variable’s measurement does not exist.
units	yes	A CF Units-compatible string [16] indicating the units the data are measured in.
valid_delta	no	Largest difference between any two consecutive values in the data above which should be treated as missing. If applying QC tests, values exceeding this delta should be given a “Bad” assessment and be replaced with _FillValue.
valid_range	no	Two-element array of [min, max] outside of which the data should be treated as missing. If applying QC tests, values outside of this range should be given a “Bad” assessment and be replaced with _FillValue.
warn_delta	no	Largest difference between any two consecutive values in the data above which should be treated with skepticism. If applying QC tests, values exceeding this delta should be given an “Indeterminate” assessment.
warn_range	no	Two-element array of [min, max] outside of which the data should be treated with skepticism. If applying QC tests, values outside of this range should be given an “Indeterminate” assessment.

As described above, this section provides only the minimal set of provenance metadata that should be captured. **If a relevant content model applies to the data (see Section 5),**

additional attributes defined by the content model should also be captured. Users may also add any additional variable attributes required to adequately describe their data.

4.3.3 Coordinate Variables

A variable with the same name as a dimension is called a coordinate variable. Coordinate variables contain the coordinate values for a dimension. A coordinate variable may not have any _FillValue or NaN (not a number indicator) values and must be monotonically increasing or decreasing. Coordinate variables should include the required long_name and units attributes. Examples of typical coordinate variables include time, bin, height, range, or depth, corresponding to dimensions with the same names.

4.3.3.1 The time Variable

Time in processed data files must be either increasing or decreasing and may not repeat. The time variable must not have any _FillValue or NaN values in any file produced by the ME data pipelines, except for Raw files. Files failing these requirements will be logged for manual review by the pipeline developer or instrument mentor.

The time variable follows CF convention which specifies time to have a very specific units format as described by https://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/build/ch04s04.html [15]. The default time zone is Coordinated Universal Time (UTC), but a different time zone may be defined using a time zone offset from UTC. Time is a “coordinate variable” or a variable with the same name as the time dimension. For time series data, time will always be specified by a coordinate variable named “time” which corresponds to the time dimension. For example:

```
dimensions:
       time = UNLIMITED ; // (1440 currently)

variables:

       double time (time);
              time:long_name = “Time offset from epoch”;
              time:units = “seconds since 1970-01-01T00:00:00”;
              time:standard_name = “time”;
```

4.3.4 Location Variables

Location variables are used to specify the geospatial coordinates of the location where an instrument resides. The instrument location is described using a latitude, longitude, and vertical distance variable. If the instrument is located above ground, then the vertical distance is specified via an altitude variable. If the instrument is located below the surface, then the vertical distance may be specified by a depth variable or a height variable, depending upon the frame of reference.

For instruments located above the surface:

- The required unit of latitude is degrees north, with a variable name of latitude, and standard_name = “latitude.”
- The required unit of longitude is degrees east, with a variable name of longitude, and standard_name = “longitude.”

- The recommended unit of altitude is “meters above mean sea level” with a variable name of altitude and standard name = “altitude.”

The altitude measurement references the altitude of ground level relative to mean sea level. To compute the instrument height above-ground level, the altitude is added to the sensor_height attribute.

Example:

```
float latitude;
       long_name = “North latitude”;
       units = “degree_N” ;
       standard_name = “latitude” ;
       valid_range = [-90.f, 90.f] ;

float longitude;
       long_name = “East longitude”;
       units = “degree_E”;
       standard_name = “longitude”;
       valid_range = [-180.f, 180.f] ;

float altitude;
       long_name = “Altitude above mean sea level”;
       units = “m”;
       standard_name = “altitude”;
```

Note the use of valid_range attributes with the latitude and longitude variables. These are not quality control limits, but limits of acceptable values to be used by the data user. A value outside these ranges should not be used and may be excluded automatically by APIs when reading from NetCDF.

For instruments located below the surface:

Latitude and longitude variables will be specified as defined for above-ground instruments. *This document will be updated as recommended standards for **depth** variables are established by the community.*

4.3.5 Data Quality

Data produced by ME pipelines can include self-documenting metadata to describe any QC checks and/or quality assurance corrections that have been applied to the data as part of the pipeline.

As part of data processing, QC tests can be applied to evaluate the quality of a variable’s data. Sometimes when QC checks are applied, it is desirable to flag whether the data passed or failed the test but leave the values unchanged. This can be useful for example, to flag when a value passes outside a range of concern to indicate it may be suspect. Recording the results of QC testing inside a dataset is described in Section 4.3.5.1.

In other cases, however, it is desirable to correct the values, as the data would not be usable if left unchanged. This could happen for example, if there was a duplicate timestamp. It may be desirable to remove one of the duplicate records altogether. If data corrections are made as a result of a failed QC test, they should be recorded as described in Section 4.3.5.2.

4.3.5.1 Quality Control

To document the results of quality control checks performed on a variable's data, optional QC variables may be added to a dataset. QC variables are called companion or ancillary variables, as they are always associated with a corresponding data variable and include the results of quality control tests performed on that variable. *Note that QC variables are not typically associated with coordinate variables as problems with coordinate variables should generally result in either a pipeline failure or a quality assurance correction via code.*

A companion QC variable will have the same dimensionality and name as the original variable field with the addition of a "qc_" prepended to the field name. For example, the QC variable for a ***temperature*** variable would be stored in a new field called ***qc_temperature***. The QC variable is linked to the data variable by adding an ancillary_variables attribute to the data variable with the value equal to the QC variable name. (Note that multiple ancillary variables may be listed for a given data variable.) For example:

```
dimensions:
       time = UNLIMITED ; // (1440 currently)

variables:

       float atmos_pressure(time)
              long_name = Atmospheric pressure
              units = kPa
              ancillary_variables = qc_atmos_pressure
```

QC Variable Attributes

Metadata describing the specific quality control tests performed on a data variable is contained in attributes on the companion QC variable. Table 3 describes the required metadata for all QC variables.

Table 3. Required and Recommended QC Variable Attributes

QC Attribute Name	Required?	Description
_FillValue	yes	_FillValue must be set to "0".
comment	yes	Must be set to "This variable contains bit-packed integer values, where each bit represents a QC test on the data. Non-zero bits indicate the QC condition given in the description for those bits; a value of 0 (no bits set) indicates the data has not failed any QC tests."
flag_assessments	yes	Array of strings describing the outcome of each QC test. Each outcome assessment must be either "Bad" or "Indeterminate".
flag_masks	yes	Array of integer values representing the bit flagged for each respective QC test performed. See section 4.3.5.2 for more information.
flag_meanings	yes	Array of strings describing each QC test performed.
long_name	Yes	Long name must be set to "Quality control flags for variable: XXX", where XXX is the name of the variable this QC variable applies to.
standard_name	yes	Standard name must be set to "quality flag".
units	yes	Units must be set to "1".

Example:

```
float qc_atmos_pressure(time)
       long_name      = "Quality control flags for variable: atmos_pressure"
       standard_name  = "quality_flag"
       units          = 1
       _FillValue     = 0
       flag_masks     = [1, 2, 4, 8, 16]
       flag_assessments = ["Bad", "Bad", "Bad", "Indeterminate",
                           "Indeterminate"]
       flag_meanings  = ["Value is equal to _FillValue", "Value is less than
                           fail_range", "Value is greater than fail_range",
                           "Value is less than warn_range", "Value is greater
                           than warn_range"]
       comment        = "This variable contains bit-packed integer values, where
                           each bit represents a QC test on the data. Non-zero
                           bits indicate the QC condition given in the
                           description for those bits; a value of 0 (no bits
                           set) indicates the data has not failed any QC tests."
```

Bit-packed Values

In order to store multiple QC flags for a given data point in a single variable, QC variable values will be stored as a bit packed, 32-bit integer. As digital information is stored as a series of bits (ones and zeroes), a 32-bit integer for the number 41 would look like this in digital form:

```
0000 0000 0000 0000 0000 0000 0010 1001
```

Bit packing involves using each individual bit comprising the integer value to represent a QC flag, where:

- Each bit represents a single QC test
- A bit value of 0 means pass, 1 means fail
- Integer values of 0 mean all QC tests passed for the given field

A flag mask is an integer that is used to determine if a particular bit has been set in a QC variable value. For example, the integer above (41) has failed tests 1, 4, and 6. To find out if test 6 failed, we could apply the flag mask corresponding with test 6 (32) to the QC variable value via a bit-wise AND operation. If the result is non-zero, then the test has failed. This operation is shown below:

```
0000 0000 0000 0000 0000 0000 0010 1001  (41) ← QC Variable value
                    0010 0000  (32) ← Flag mask for test 6
-------------------------------------------
                    0010 0000  (32) != 0 ← Test 6 failed
```

4.3.5.2 Quality Assurance

In the case where a failed QC test results in a correction applied to a variable's data, this correction should be documented in the corresponding corrections_applied variable attribute for each variable that was affected.

5.0 Content Model-Based Dataset Requirements

In addition to the general dataset requirements defined in Section 4, specific content model requirements may also apply depending upon the type of dataset being produced. **When developing an ME pipeline dataset, users should review the MHKDR content model list at https://mhkdr.openei.org/models/ to see if one of those models applies to their data.** If a content model does apply, users should download the content model template from the MHKDR site and make sure that all specified variables, units, corrections/algorithms, and metadata are included in their pipeline so that the output dataset is compliant with the content model. **We recommend that the internal standards committee also be responsible for evolving these content models and making sure that they conform to International Electrotechnical Commission standards and domain best practices.**

6.0 Standard Names

Standard names represent the “official” name of a variable and are used so that variables can be easily combined and cross-referenced across multiple heterogeneous datastreams, even if the variable name differs. A database of climate domain-specific names and descriptions is being maintained by Unidata CF [11]. As the Unidata list includes many oceanographic measurements, it is strongly recommended that the ME community review this list to see if it will suffice for data produced by ME data pipelines or if additional definitions are warranted. If the latter, we suggest that the ME community maintain a supplemental list of standard variable names for the marine renewable energy domain. This document will be updated to reflect any changes in community policy.

7.0 Data Levels

ME data pipelines will use data levels to indicate the “level of processing” of a specific datastream/dataset. The lowest level indicates raw data, and each subsequent data level indicates that further quality assurance and/or algorithms have been applied. A data level consists of one lowercase letter followed by one number (except for raw data) and will be included as part of the file naming scheme described in Section 5. This section describes the standard data levels that will be produced by ME pipelines as shown in Figure 2.

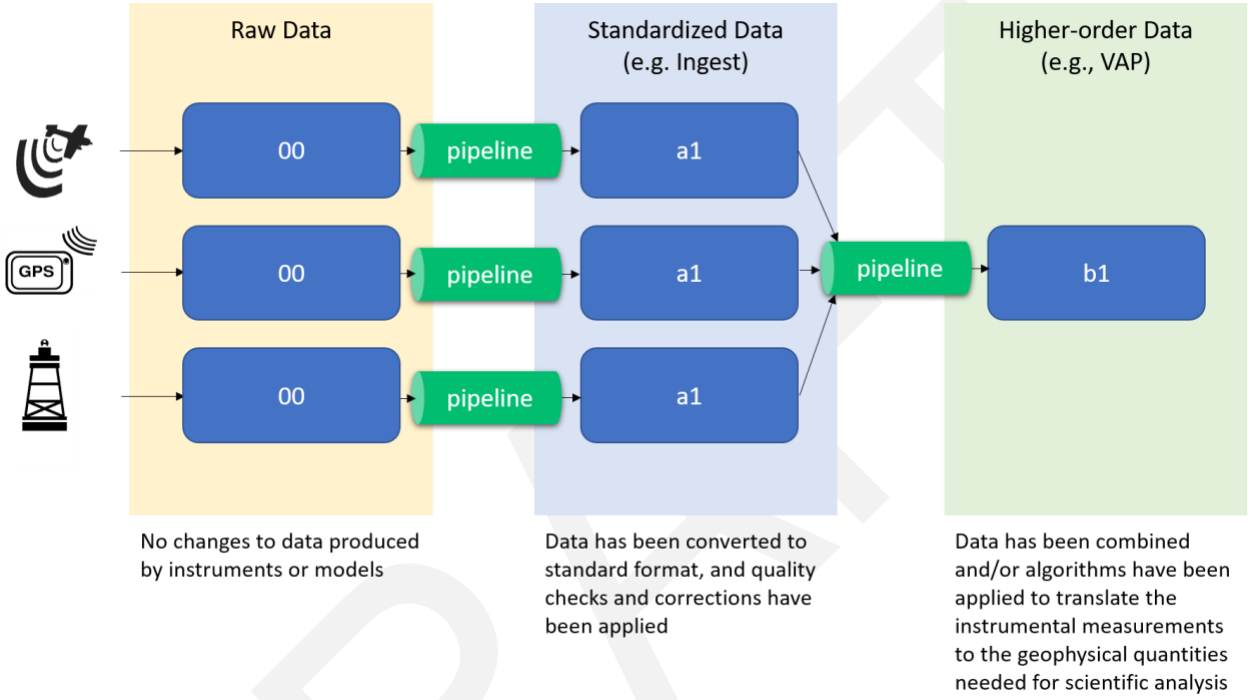

Figure 2. Data Levels

Standard Data Levels

00: raw data − Primary raw data collected directly from the instrument. All ingest pipelines will store the original raw data iif subsequent review of processing algorithms is required.

a1: standardized data − Raw data that has 1) been converted to standard file format, 2) been converted to standard engineering units, 3) had standard corrections for the instrument/measurement or calibrations applied, and 4) had quality control checks and/or cleaning applied.

b1: higher order data – Derived data computed by applying algorithms to one or more datastreams to translate instrumental measurements into the geophysical quantities needed for scientific analysis. b1-level data may be transformed onto a different temporal resolution, may be combined with other external data, and may have domain-specific algorithms applied.

Other Data Levels

Data producers have the discretion to increment the second digit on the data level to reflect special circumstances or cases where multiple rounds of processing have been performed.

01 to 09: Raw data levels above 00 indicate that the original source data had already been pre-processed by external providers. An example would be if an external dataset from the MHKDR was standardized via a ME data pipeline. (It is recommended that original instrument/model data be used whenever possible for traceability purposes.)

a2 to a9: Ingest data levels above a1 indicate that further processing was performed to add additional calibration factors, cleaning, or quality tests after the data had already been processed as an a1 datastream. A description of the further processing must be included in the global attributes for the dataset.

b2 to b9: Data levels above b1 indicate that further processing was applied to a b1-level datastream using the same temporal resolution. Possible reasons for increasing levels include better calibration, better coefficients for algorithms, or reprocessing using different averaging resolution algorithms.

8.0 File Naming Conventions

File naming conventions are an integral component of the ME data processing pipeline, as it allows scripts to automate the process of retrieving, extracting, and exporting ME data files with minimal overhead in memory and processing. File names are intended to be self-describing and include descriptors for the instrument and location where the data were captured, the data level, and the time at which the data starts. The naming conventions for the different types of files produced by ME data pipelines are detailed below.

8.1 Naming Conventions for Processed Data Files

All processed data output by ME data pipelines will be named according to the following format:

(location_id).(dataset_name)[-qualifier][-temporal].(data_level).(date).(time).(ext)

Table 4. Components of File Naming Scheme

Part of Filename	Source	Description
location_id	Global attribute	Mandatory string used to identify the location where the data were obtained.
dataset_name	Global attribute	Mandatory string used to identify the type of data being produced. Must not end in a number.
qualifier	Global attribute	Optional qualifier used to distinguish this dataset from other datasets produced by the same instrument or other downstream processes. Must not end in a number.
temporal	Global attribute	Optional string used to indicate the temporal resolution of the data. Examples include, but are not limited to, 10hz, 30s, 5m, 1h, 7d, 1mo, and 1yr.
data_level	Global attribute	Mandatory two-character descriptor used to indicate the level of processing of the data. See Section 7 for more details.
date	Time variable	The UTC date in year, month, day-of-month format consisting of exactly eight characters of the first data point in the file. For example, “March 3rd, 2021” would be written as the date “20210319”.
time	Time variable	The UTC time in hour, minute, second format consisting of exactly six characters of the first data point in the file. For example, the time “2:47 p.m. and 50 seconds” would be written as “144750”.
ext	Config File	The file extension.

** Note that the “.” and “-“ characters are delimiters and may only be used in the specific places shown above*

Ingest Example:

The following filename describes data produced by a buoy stationed in Morro Bay, California taking wind measurements every 10 minutes starting at midnight on December 1st, 2020:

morro.buoy_z06-wind-10m.a1.20201201.000000.nc

ME data pipeline support for consolidating datastreams or transforming datastreams to customized time and height grids will be coming in FY 2022. At such time, this document will be updated with additional examples from the ME community.

8.2 Naming Conventions for Plots and Other Files

ME data pipelines may produce plots and other files to be used for data review purposes. In this case, accompanying plot files will follow the exact same naming conventions used for processed data files, with the addition of a "plot description" section as described below:

(location_id).(dataset_name)[-qualifier][-temporal].(data_level).(date).(time).(title).(ext)

Where **title** is a string used to describe the plot or other media file.

As an example, the following filenames describe plots produced by an ME data pipeline for a buoy stationed in Morro Bay, California taking wind measurements every 10 minutes starting at midnight on December 1st, 2020:

morro.buoy_z06-wind-10m.a1.20201201.000000.wind_vectors_by_height.png
morro.buoy_z06-wind-10m.a1.20201201.000000.avg_power_generation.png
morro.buoy_z06-wind-10m.a1.20201201.000000.wind_speed_animation.gif

8.3 Naming Conventions for Raw Data Files

As part of an ME data pipeline, raw data files will be renamed to follow ME data pipeline filename conventions. All raw files ingested by ME data pipelines will be renamed according to the following format:

(location_id).(dataset_name)[-qualifier][-temporal].(data_level).(date).(time).raw.(original_filename)

As an example, the following filename describes data produced by a buoy stationed in Morro Bay, California where the filename that came straight from the instrument was buoy.z06.20201201.000000.csv:

morro.buoy_z06-wind-10m.00.20201201.000000.raw.buoy.z06.20201201.000000.csv

9.0 References

1. MHKiT: https://github.com/MHKiT-Software
2. http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html
3. https://www.arm.gov/publications/programdocs/doe-sc-arm-15-004.pdf
4. 2019_QARTOD_Waves_Update2Final_Feb2019.pdf
5. 2019_QARTOD_Currents_Update_Second_Final_July2019.pdf
6. http://www.oceandatastandards.org/
7. http://www.marinet2.eu/wp-content/uploads/2017/04/D2.09-Standards-for-Wave-Data-Analysis-Archival-Presentation.pdf
8. XArray: http://xarray.pydata.org/en/stable/
9. NetCDF: https://www.unidata.ucar.edu/software/netcdf/
10. Parquet: https://parquet.apache.org/
11. CF Standard Names: http://cfconventions.org/standard-names.html
12. Udunits: https://www.unidata.ucar.edu/software/udunits/udunits-current/doc/udunits/udunits2.html
13. Content Standard for Digital Geospatial Metadata: https://www.fgdc.gov/standards/projects/metadata/base-metadata/v2_0698.pdf
14. CF Conventions for cell_methods: http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#appendix-cell-methods
15. CF Conventions for time: https://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/build/ch04s04.html
16. CFUnits: https://ncas-cms.github.io/cfunits/cfunits.Units.html
17. cfunits: https://pypi.org/project/cfunits/
18. cftime: https://pypi.org/project/cftime/

Appendix A – MHKDR Taxonomy

Technologies

- Current
  - Axial Flow Turbine
  - Cross Flow Turbine
  - Oscillating Hydrofoil
  - Tidal Kite
  - Archimedes Screw
- Wave
  - Attenuator
  - Point Absorber
  - Pressure Differential
  - Oscillating Water Column
  - Overtopping
  - Surge Converter
  - Terminator
- OTEC
  - Closed Cycle
  - Open Cycle
  - Hybrid
- Salinity Gradient
  - Pressure Retarded Osmosis
  - Reverse Electrodialysis

Topics

- Economics
  - LCOE
  - Cost Assessment
  - Supply Chain
- Environmental
- Resource
- Technology

Collection Method

- Field Data
- Lab Data
- Modeling
- Test Center

Appendix B – NetCDF Ingest File Example

Filename: `$ROOT/morro/ morro.buoy_z06.b1.20201201.000000.nc`

Global Attributes:

```
code_url = "https://github.com/clansing/tsdat/releases/tag/1.0"
conventions = "ME Data Pipeline Standards v1.0"
data_level = "b1"
description = "Pacific Northwest National Laboratory (PNNL) manages this AXYS
        WindSentinelTM buoy (Buoy #130) on behalf of the U.S. Department of Energy (DOE)
        that collect a comprehensive set of meteorological and oceanographic (metocean)
        data to support resource characterization for wind energy offshore. The buoy
        collects in-situ sea surface temperature, salinity, ocean currents, and wave data
        as well as near-surface air temperature, humidity, and horizontal wind speed and
        direction."
dataset_name = "buoy_z06"
instrument_manufacturer = "AXYS Technologies Inc."
instrument_meaning = "Self-powered floating buoy hosting a suite of meteorological and
        marine instruments."
instrument_name = "WindSentinel"
location_id = "morro"
location_meaning = "Morro Bay, CA"
title = "Surface Meteorological Measurements and Ocean Current Measurements from Buoy
        #130 at Morro Bay, CA"
history = "Ran at 2021-03-21 18:41:20"
datastream = "morro.buoy_z06.b1"
input_files =
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.currents.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.conductivity.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.wind.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.rh.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.gill.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.temperature.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.gps.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.surfacetemp.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.pressure.csv",
"morro.buoy_z06.00.20201201.000000.raw.buoy.z06.00.20201201.000000.pyranometer.csv"
```

Dimensions:

```
time = UNLIMITED  // (1440 currently)
depth = 50
```

Variables:

```
int64 time (time)
    long_name = "Time (UTC)"
    standard_name = "time"
    units = "seconds since 1970-01-01"
    calendar = "proleptic_gregorian"

int64 depth (depth)
    long_name = "Depth"
    units = "m"

double current_speed (time, depth)
    _FillValue = -9999.
    long_name = "Current Speed"
    units = "mm/s"
    fail_range = [0, 10000]
    ancillary_variables = "qc_current_speed"

double current_direction (time, depth)
    _FillValue = -9999.
    long_name = "Current Direction"
    units = "degrees"
    fail_range = [0, 360]
    ancillary_variables = "qc_current_direction"

double sea_surface_temp (time)
```

```
    _FillValue = -9999.
    long_name = "Sea Surface Temperature"
    units = "degC"
    fail_range = [-5, 50]
    ancillary_variables = "qc_sea_surface_temp"

double latitude (time)
    _FillValue = -9999.
    long_name = "Latitude"
    units = "degree_N"
    ancillary_variables = "qc_latitude"

double longitude (time)
    _FillValue = -9999.
    long_name = "Longitude"
    units = "degree_E"
    ancillary_variables = "qc_longitude"

int qc_current_speed (time, depth)
    long_name = "Quality check results on field: Current Speed"
    units = "1"
    flag_masks = [1, 2, 4]
    flag_meanings = ["Value is equal to _FillValue or NaN",
               "Value is less than the fail_range",
               "Value is greater than the fail_range"]
    flag_assessments = ["Bad", "Bad", "Bad"]
    standard_name = "quality_flag"

int qc_current_direction (time, depth)
    long_name = "Quality check results on field: Current Direction"
    units = "1"
    flag_masks = [1, 2, 4]
    flag_meanings = ["Value is equal to _FillValue or NaN",
               "Value is less than the fail_range",
               "Value is greater than the fail_range"]
    flag_assessments = ["Bad", "Bad", "Bad"]
    standard_name = "quality_flag"

int qc_sea_surface_temp (time)
    long_name = "Quality check results on field: Sea Surface Temperature"
    units = "1"
    flag_masks = [1, 2, 4]
    flag_meanings = ["Value is equal to _FillValue or NaN",
               "Value is less than the fail_range",
               "Value is greater than the fail_range"]
    flag_assessments = ["Bad", "Bad", "Bad"]
    standard_name = "quality_flag"
```

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354
1-888-375-PNNL (7665)

www.pnnl.gov