ANDE Format Specification Version 0.2.0 Preliminary

Stephen D. Holland and others

April 4th, 2023

1 Introduction

2 Concepts

The basic concept of the ANDE format is the "recording". Conceptually, the recording is some unit of acquired and/or processed data that you are storing. Recordings can be large or small, representing as little as a single number (or even be entirely empty), or multiple grouped datasets, gigabytes or even terabytes in size.

Most NDE data is naturally stored in an array. For example, a digitally-sampled ultrasonic A-scan representing pressure as a function of time, is naturally represented by the stored sample values in an array where the index maps to time. However, just storing the sample values is not sufficient, as you don't know the initial time, the time step, or how to interpret the sample values.

By storing the NDE data in ANDE format, you can include metadata such as the time step, coordinate axis specifications, and units. You can also group multiple arrays by name within the same file, and even (if desired) have a tree of recordings, like files within folders on your computer.

More sophisticated acquisition can result in higher-dimensional data, such as C-scans, full matrix capture ultrasound, and X-ray computed tomography. All of these modalities generate multi-dimensional arrays that are stored in ANDE format as easily as the single A-Scan.

The ANDE format was also designed to be readily extensible to support additional data types and representations including surface data from CAD models, kinematic models for robotic systems, and 3D geometry more generally.

The specification so far focuses on very basic classes. However, it includes class tag functionality that is intended to support grouped metadata for common applications, such as transducer specification and similar. The presence of a particular class tag will indicate to the reader that metadata conforming to a corresponding specification is present. In addition, it is anticipated that new classes will be defined in the future to correspond to specific NDE measurement scenarios for specific NDE modalities.

3 Data Model and HDF5 Representation

The overall structure of ANDE data is a hierarchical tree of groups containing recordings. Each recording could be an ande_group, representing a deeper level of the tree, or an ande_array. Other such classes are

also possible, including both subclasses of ande_group and ande_array, as well as entirely new data structures, which only derive from ande_recording.

The basic unit is the ande_recording, which contains primarily name and metadata. The ande_group enables the tree structure. The ande_array contains array-structured data. Since ande_array and ande_group both inherit from ande_recording, they also can contain metadata.

ANDE metadata is attached to ande_recording. It consists of multiple named entries. The entry names are generally prefixed by the name of the class or class tag which specifies the entry. Metadata can have string, double-precision floating point, signed and unsigned 64-bit integer, and boolean data types.

The root of the data structure is always a group with a blank name. This group is refered to with the path "/". Recordings can be identified via slash-delimited paths, similar to the path portions of web URLs, or POSIX file paths. For example, "/ultrasound_test/Cscan" would refer to the "Cscan" recording within the "ultrasound_test" group within the root group.

The primary storage layer is HDF5, and in the HDF5 representation, the various classes are represented as HDF5 groups containing specific HDF5 attributes, HDF5 datasets, and/or HDF5 subgroups. Note that the ANDE paths mentioned above are different from HDF5 paths; for example the ANDE path "/ultrasound_test/Cscan" would have an HDF5 path of "/ande_group-subgroups/ultrasound_test/ande_group-subgroups/Cscan" (per the ande_group specification below).

HDF5 datasets are used for storing the array data, and array dimensions. Within the HDF5 context, array data is stored as a unidimensional array, along with a companion integer dimension dataset (dimlenC or dimlenF) that indicates the lengths of the axes, and whether the data is stored C-style (row major with the last index changing most rapidly), or Fortran-style (column major with the first index changing most rapidly). Data layouts other than contiguous C-style or contiguous Fortran-style are not supported.

Metadata is supported as named HDF5 attributes attached to an HDF5 subgroup. Boolean metadata is stored as an H5T_ENUM within an H5T_NATIVE_UINT8, with two possible values: FALSE (0) and TRUE (1).

Strings are stored as variable length null terminated H5T_STRING with character type H5T_C_S1 and UTF8 character set.

Non-HDF5 representations of ANDE data are also possible, such as JSON serializations of the tree structure. Such forms would usually be used to provide dual compatibility files between ANDE and some other format. No such forms have yet been formally defined.

3.1 ande_recording

The ande_recording class is the basic element of the ANDE data model. It defines a unit of information that has a name and that can have arbitrary metadata attached.

Class `ande_recording` version 0.2.0

Name	HDF5 Type	Value Type	Require	Description
`ande_classes`	Attribute	String array	Must contain	Names of derived class and all ancestor classes; at minimum ande_recording.
`ande_class-tags`	Attribute	String array	May contain	Tagged characteristics, usually specified in metadata.
`ande_recording-label`	Attribute	String	Must contain	The label of this recording within its parent group.
`ande_recording-version`	Attribute	String	Must contain	A version of the ande_recording specification that this recording is compatible with.
`ande_recording-metadata`	Group	Group	Must contain	Named metadata entries as hdf5 attributes. Attribute values can be strings, floating point, or signed or unsigned integers, or booleans. Booleans are represented as an hdf5 enumeration with two values, 0 and 1. Attribute names beginning with 'ande_' are reserved for standardized attributes.

3.2 ande_group

The ande_group class is an nde_recording that allows nested recordings (composite design pattern). Entries within an ande_group are generally indexed by name, not by an ordering within the group. If ordering is important, name the entries so that alphanumeric sorting will result in the correct order. The underlying HDF5 library can be configured to track ordering, but this is not currently used, and other storage layers may not.

Class `ande_group` version 0.2.0 ; derives from ande_recording

Name	HDF5 Type	Value Type	Require	Description
`ande_classes`	Attribute	String array	Must contain	Names of derived class and all ancestor classes; at minimum ande_recording and ande_group.
`ande_class-tags`	Attribute	String array	May contain	Tagged characteristics, usually specified in metadata.
`ande_recording-label`	Attribute	String	Must contain	The label of this recording within its parent group.
`ande_recording-version`	Attribute	String	Must contain	A version of the ande_recording specification that this recording is compatible with.
`ande_recording-metadata`	Group	Group	Must contain	Named metadata entries as hdf5 attributes. Attribute values can be strings, floating point, or signed or unsigned integers, or booleans. Booleans are represented as an hdf5 enumeration with two values, 0 and 1. Attribute names beginning with 'ande_' are reserved for standardized attributes.
`ande_group-version`	Attribute	String	Must contain	A version of the ande_group specification that this recording is compatible with.
`ande_group-subgroups`	Group	Group	Must contain	Zero or more hdf5 subgroups each containing an ande_recording, named according to their labels.

3.3 ande_array

The ande_array class is an ande_recording that stores one or more multidimensional arrays. In general, it will store exactly one multidimensional array with corresponding metadata defining the coordinate axes, step sizes, and units. In almost all cases, multiple arrays should be stored as separate ande_array instances within a containing ande_group.

Class ande_array version 0.2.0 ; derives from ande_recording

Name	HDF5 Type	Value Type	Require	Description
ande_classes	Attribute	String array	Must contain	Names of derived class and all ancestor classes; at minimum ande_recording and ande_array.
ande_class-tags	Attribute	String array	May contain	Tagged characteristics, usually specified in metadata.
ande_recording-label	Attribute	String	Must contain	The label of this recording within its parent group.
ande_recording-version	Attribute	String	Must contain	A version of the ande_recording specification that this recording is compatible with.
ande_recording-metadata	Group	Group	Must contain	Named metadata entries as hdf5 attributes. Attribute values can be strings, floating point, or signed or unsigned integers, or booleans. Booleans are represented as an hdf5 enumeration with two values, 0 and 1. Attribute names beginning with 'ande_' are reserved for standardized attributes.
ande_array-version	Attribute	String	Must contain	A version of the ande_array specification that this recording is compatible with.
ande_array-numarrays	Attribute	Integer	Must contain	The integer attribute representing the number of multi dimensional arrays stored within this ande_array. This is 1 in almost all cases, because usually when multiple data arrays are needed they should each have their own ande_array.
For each multi dimensional array, indexed i starting from zero:				
ande_array-name-<i>	Attribute	String	Must contain	The name of the array corresponding to index i. Indices start from zero. If no other naming is required, the single array within the ande_array should be named "array-0". The index must not contain leading zeros.
ande_array-array-<i>	Dataset	Array	Must contain	An HDF5 dataset containing the data for the array corresponding to index i. Indices start from zero. The data should be contiguous and stored in either C order (row major) or Fortran order (column major). ande_array-array-<i> should have an HDF5 string attribute named ande_array-nativetype containing a string representation of the appropriate hdf5 native type for the array data. Specifically, "H5T_NATIVE_FLOAT" for 32-bit floating point, "H5T_NATIVE_DOUBLE" for 64-bit floating point, "H5T_NATIVE_INTx" for an x-bit signed integer, or "H5T_NATIVE_UINTx" for an x-bit unsigned integer.
ande_array-dimlenC-<i>	Dataset	Integer array	Must contain either this or dimlenF	Specifies that the array is stored in C order and contains an HDF5 dataset containing the array dimensions for the array corresponding to index i.
ande_array-dimlenF-<i>	Dataset	Integer array	Must contain either this or dimlenC	Specifies that the array is stored in Fortran order and contains an HDF5 dataset containing the array dimensions for the array corresponding to index i.

The ande_array class also specifies metadata representing axis information for the different axes of the array. Even if the ande_array contains multiple multi dimensional arrays, only one set of axis information is provided, and this information implicitly relates to the 1st (index zero) array. This metadata is stored in the ande_recording-metadata subgroup defined above.

Metadata for class `ande_array` version 0.2.0

Name	Value Type	Require	Description
`ande_array-ampl_coord`	String	Should contain	A string representing the name of the coordinate axis corresponding to the numbers stored in the array. If missing, then the coordinate is interpreted as a default coordinate of "Voltage".
`ande_array-ampl_units`	String	Should contain	A string representing the units of the coordinate axis corresponding to the numbers stored in the array, once multiplied by the scaling factor and added to the offset. If missing, then the units are interpreted as a default unit of "Volts".
`ande_array-ampl_offset`	Double	Should contain	An offset to be added post-scaling to the numbers stored in the array. If missing, the offset is interpreted as 0.0.
`ande_array-ampl_scale`	Double	Should contain	A scale to be multiplied pre-offset to the numbers stored in the array. If missing, the scale is interpreted as 1.0.
For each axis j of the multi dimensional array, the following metadata entries should be defined:			
`ande_array-axis<j>_coord`	String	Should contain	A string representing the name of the coordinate axis. If missing, then the coordinate is interpreted as a default coordinate of "Time".
`ande_array-axis<j>_offset`	Double	Should contain	A floating point number representing the coordinate of the first entry along this axis in physical units. If missing, the value is interpreted as 0.0.
`ande_array-axis<j>_offset-units`	String	Should contain	A string representing the units for `ande_array-axis<j>_offset`. This and `ande_array-axis<j>_scale-units` must be specified (or not specified) as a matching pair. If missing, the units are interpreted as "seconds".
`ande_array-axis<j>_scale`	Double	Should contain	A floating point number representing the step size along this axis in physical units. If missing, the value is interpreted as 1.0.
`ande_array-axis<j>_scale-units`	String	Should contain	A string representing the units for `ande_array-axis<j>_scale`. This and `ande_array-axis<j>_offset-units` must be specified (or not specified) as a matching pair. If missing, the units are interpreted as "seconds".

4 Discussion Topics

- scaling/probe attenuation – add offset and scale metadata entries
- ordering within a group – agreed that order is not significant, except perhaps alphanumeric order
- storage of C-scan data; relationship of raw data cube, extracted C-scans, and time gates/thresholds – separate storage of data cube, configured C-scans, and pre-extracted C-scans

- encoding kinematic model(s) along with measured data to facilitate CAD model projection. – store coordinate frames and parametric transforms, tagged to specify at least one specific path that connects specimen CAD model and sensor ray. Multiple paths are possible and multiple data sources for post-processing refinement. Use Gaussian mixtures as a general form of uncertainty representation to facilitate post-processing, for example by Monte Carlo methods