Development of a Long Term Viable Dynamic Data Archive Format

Allyn W. Phillips, PhD, Research Associate Professor
Randall J. Allemang, PhD, Professor
School of Dynamic Systems, College of Engineering and Applied Sciences
PO Box 210072, University of Cincinnati, Cincinnati, OH 45221-0072 USA

ABSTRACT
For nearly forty years, the Universal File Format has served as a de facto standard for cross platform data interchange and archiving. However, as technology has progressed, the aging nature of this eighty character ASCII FORTRAN card image based format has become problematic. As a result, with the ever increasing legal requirements of long term record keeping, a flexible, open definition file format suitable for viable long term archiving of data and results, which is not dependent upon any particular hardware or operating system environment has become necessary. This paper focuses upon the various (sometimes conflicting) issues involved in the decision process and the resulting principal identified features necessary for realistic, long term reliable recovery of information and successful community adoption.

1. Introduction

1.1 DSA Objective

Within the vibration technical community, there is a need for a long term viable, open definition file format for the archiving of dynamic signal data and results. This flexible archive format, independent of any particular hardware or operating system environment and distinct from any particular database management structure, is needed in order to satisfy the increasing legal requirements of long term record keeping. For many years, the Universal File Format has been the de facto standard in this area. However, as technology has progressed, the aging nature of this eighty character line oriented, ASCII, FORTRAN card image based format has become problematic. Following a brief discussion of some of the strengths and weaknesses of existing data formats, this document focuses upon the identified feature set needed for realistic, long term reliable recovery of information and successful future community adoption.

Additionally, as a direct outgrowth of the long term archive integrity objective and the ever increasing capacity of data storage media, the opportunity for an important paradigm shift from the traditional storage of reduced frequency data to the storage of complete raw time series becomes viable. The long term preservation of the underlying original data sources provides the capability to reanalyze the acquired test data at some unknown future date. Such unforeseen analyses and data mining exercises are anticipated in the event of unexplained system behavior and/or as more sophisticated data reduction algorithms are developed.

1.2 Guiding Principals

It should be noted that, for the purposes of discussion, the existing Universal File Format (UFF) has been taken as the initial starting reference point. Various other formats were reviewed and considered for content and applicability; however, in order to facilitate technical community adoption, the final resulting format has been specified to be open, extensible, and non-proprietary. In addition, the principle of 'keeping it simple' has been followed in order to facilitate both industry acceptance and long term comprehension. The recognition is that if it gets too complicated, no one will use, support, or adopt it. For this reason, the final resulting format will probably not be perfect for everyone but it should sufficient for everyone's needs, in other words, a 95% solution. This decision is consistent with the consensus of opinion expressed at a meeting of users and vendors held at IMAC in 1998. The focus of that meeting was to determine the interest level and collect ideas for extending the UFF to address some of its basic deficiencies. In many respects, this project has benefited from and is somewhat of an outgrowth of that activity.

Before discussing the new format, it is important to avoid initial misconceptions by discussing briefly what the new format is not. The new format is focused upon long term archival of dynamic data; as such, issues like the data storage media (hardware) and the vendor specific internal database structures are not being addressed. There is no intention or desire to force any particular hardware or internal database structure upon individual vendors or users. The only goal is to produce a long term, viable, cross platform, open architecture, dynamic data storage format. In practical terms, this format should allow users to export data into a data archival structure that is independent of computer operating system and/or the original application program that generated the data and ultimately retrieve the data into other operating systems and other application programs at some later date (up to 50-75 years later, if necessary). The realistic need to move the data archive from one form of data storage media to another over this extended period of time is not a concern of this effort.

It is also important to recognize that the since the content focus is dynamic data, other data content types, such as CAD/CAE, video, pictures, etc., will not be specifically included in the format. It should be noted, however, that although such data will not be specifically identified and targeted for support, nothing in the definition will prevent referencing such information via the metadata records or including it within the data archive container.

One overarching principal however is recoverability! As a long term archive format, any feature or suggestion which jeopardizes recoverability must be subservient. One example of this is the decision to abandon strict backward compatibility with the existing UFF definition; instead, to handle UFF as well as other data formats via an importer/convertor.

1.3 Long Term Goals

The UFF was developed in the late 1960's by the Structural Dynamics Research Corporation (SDRC). The original intent was as a cross platform (software) interchange format. It functioned well in that capacity and because of its success it became the default archive format.

The advantage is that the UFF has been relatively stable and effective for around 40 years. While nobody particularly likes it, nonetheless, as a least common denominator, it has in the past basically gotten the job done. What is needed is to address the core UFF weaknesses that have developed over the years as technology has advanced. The DSA format is intended to fundamentally extend and/or replace the UFF; hence it might be thought of as roughly 'UFF2ish', a sort of second generation UFF.

Again, another important point of clarity should be noted: the purpose of the format is primarily archival, not an active database. As a result, the focus of the definition is upon an archive (streamed) format, NOT upon any particular programming language implementation or representation. Retrieval performance of the data is also NOT a primary concern. (Although as computers get faster, discs get bigger, and memory gets cheaper, the issue of adequate performance should be moot.) As an archival format, there is no particular emphasis upon any particular performance in random access (read) or upon necessarily even supporting random access (write) capability.

The long term goal is to encourage adoption by the dynamics community (both vendor and user) as an export and import format by having a set of libraries in both source and executable format on the University of Cincinnati - Structural Dynamics Research Laboratory (UC-SDRL) web site for use by the community. The UC-SDRL web site will provide a clearing house for enhancements and bug fixes which can be submitted back to UC-SDRL for

incorporation into the reference implementations. Currently, the UC-SDRL web site provides documentation for the existing UFF data structures.

Finally, it is the intention that long term there will be a set of software test suites to facilitate compliance and validation checking of implementations. There is no intention to require the community (and vendors in particular) to use the reference implementations in order to achieve compliance. Anyone may develop an optimized version from the specification and validate against the compliance test suite.

1.4 Historical Weaknesses & Abuses of the UFF

One of the primary UFF weaknesses is in the area of metadata where there is no well-understood mechanism for users to attach arbitrary, test relevant condition information or other pertinent comments to UFF data records in a portable manner. The desired format must include a naturally extensible metadata capability by providing mechanisms for easy, natural extension as new needs develop, while providing backward compatibility, as much as practical. Another known weakness is the aging, eighty ASCII character, FORTRAN card image format. Still another weakness is the serial stream dependency in the UFF definition. This is most serious in the area of units handling, where a loss or error can cause all succeeding records to be misinterpreted.

Over the years, because of misunderstanding of the Universal File Format definition and because of the uncertainty about handling various data types, there has arisen several frequent and yet understandable abuses of the UFF which cause the files to be less portable than they might otherwise be and effectively non-transportable between different hardware and software systems or even unreadable and unrecoverable. Some of the most notable problems have been:

- Storing critical, non-documentary information in textual ID lines
- Exceeding the 80 character line length limit
- Inconsistent, order dependent units issues (default SI units definition)
- Misunderstanding the format definition
- Invalid field data values and formats (frequently arising from programming language behavior differences [e.g. C vs. FORTRAN])
- White space errors (spaces vs. tabs)
- No clear procedure for format error handling
- Lack of user definable fields resulting in storing critical, non-documentary information in textual fields

In each of these situations, the result was a format that became less portable (at times even non-transportable) and potentially unreadable or unrecoverable.

2. History of the Project

2.1 First Year Activity

Over the course of the project, a number of different data storage formats were considered and investigated. Most of the formats, besides not meeting all the goals of the project, could not be seriously considered due to legal usage restrictions. At the end of the first phase, three potentially viable foundational solution options (or directions) which required further investigation had been identified.

The first was a DOE sponsored effort, the HDF (Hierarchal Data Format). While not specifically targeted at long-term, dynamic data archiving, it appeared to support all (or most) of the necessary features needed for such applications. The second was to use a specific vendor proprietary format as a basis of development. The third was to develop a new format from scratch based upon the needs of B&W/Y12/DOE and the aggregated vendor/user feedback.

Clearly, there was significant advantage to leveraging the work of an existing format and tailoring it to this specific application and so at that time the first option was preferred. Even so, the process of identifying and evaluating other existing data formats continued.

- Neutral Files - the Neutral File format focuses upon CAD/CAE and does not support dynamic data.
- ASAM/ODS - the thrust of ASAM/ODS is the definition of the interface into an ORACLE database.
- XML - a structured, textual information format. While not a dynamic data format, its structured organization is interesting for the metadata. (NOTE: Can contain “arbitrary binary” information through textual encoding. Typically increases file size by about 1/3.)

- Institute of Environmental Sciences and Technology (IEST): WG-DTE042: Vibration/Shock Data Storage - the WG was to focus on "establishing a standard for the storage of large binary files used mainly in vibration and acoustic testing."

Although they couldn't be considered for the basis of the format specification, several commercial databases were reviewed for content to determine the key features which needed to be supported in order to achieve acceptance among vendors.

Following the conclusion of the first phase effort and prior to the resumption in the second, a limited review of formats identified and evaluation of the feedback received continued.

2.2 Second Year Activity

Overall, the project is focused upon the long term archival of dynamic measurements and associated metadata. Performance and size of the archival are not the primary objectives; data integrity and recoverability are the prime objectives. The project is limited to the detection of inadvertent data corruption and extraction of remaining valid data. The problems associated with malicious damage are specifically outside the scope. The issue of refreshing the data, as media storage technology changes, will be required but is also not the concern of this project.
One of the challenges to this project has been that there are very few data formats that are open and usable without restriction. As the project resumed, three primary candidates which had been identified in the interim for consideration as the format basis were HDF, ASAM-ODS ATF-XML, and a custom developed XML format. Unfortunately, each had a significant weakness.

- The weakness of the HDF format is that it is essentially a binary file system embedded in a file. Long term damaged data recovery could be problematic. Also, since the two most recent versions of the format specification, HDF4 and HDF5, are incompatible, there is the additional concern that long term file compatibility could also be at risk.
- The thrust of ASAM/ODS is the definition of the interface into an ORACLE database. One of its weaknesses is that the format prefers external binary files for large data components. Using direct file references for these parts makes long term data integrity problematic. The common actions of moving or renaming files, potentially places the entire dataset at risk of complete loss.
- The weakness of a straight XML implementation is that the XML standard requires that any conforming parser must stop processing upon encountering any error. Further, extraction of any data information requires effectively reading the entire file.

Because the only historically successful, long term archival format is the traditional book, there was a focus upon ASCII/textual data type formats.

The process of reviewing the three primary archival format candidates noted above proceeded, in part, by reviewing existing available data format options with a specific focus upon applicable features for incorporation into the resultant archival format. Since most data formats are targeted at either data transport or active database manipulation, some of their design decisions are at odds with the long term archival goal of the project; nonetheless, many of their specific data features are still relevant. Some of the positive and negative aspects of these different formats were considered in light of these specific features and how long term implementation might be affected. Consideration was also given to the feature characteristics needed for long term read/recovery viability and industry acceptance. In particular, these two elements favor a format that is principally textual (ASCII) encoded data, which is nominally familiar, is simple to implement and can be mapped relatively straightforward to existing proprietary databases.

Although proprietary data formats exist which are fundamentally ASCII/binary data interleaved, such formats could not be considered because of their proprietary nature. However, one format specification standard, developed in the area of textual document interchange, appeared to have significant application to this project. It is the 'Office Open XML Format, ECMA-376, Second Edition, Dec. 2008.'

The textual document attributes of headers, footers, cross references, body text, etc. share many conceptual features common to the archiving of dynamic data, that is, data headers, metadata, cross channel references, etc. Hence, the packaging of such data can conceptually be considered a type of dynamic document. The Office Open XML Format and in particular the portion referred to as 'Open Packaging Conventions', includes many of the characteristics of the desired archive data definition. While the specification was primarily developed to support textual documents, the actual specification is general and not specific to such documents. Effectively the definition is a random access

container holding primarily textual data. By being based upon familiar industry standards (some being de facto definitions), the format has the potential for easier industry acceptance.

Thus for the second phase, the operating plan for the primary data container was to use the ECMA-376 'Open Container' (or close equivalent.) It is essentially a restricted format, industry standard ZIP file. The contents, of which, were envisioned primarily as sets of XML data streams. The strength of this container is that it can hold structurally organized data and retain the structure. It can also contain and store non-format defined (vendor specific, informational data, pictures, movies, etc.) data.

The data recovery features of the potential format are not focused upon deliberate malicious data manipulation, but upon inadvertent corruption. Depending upon the type and degree of corruption, through the use of appropriately tagged prefix metadata (linkage, checksum, et al.; effectively providing redundant container information), the valid uncorrupted data could still be extracted from a damaged archive. Thus, potentially all or most of an archive could be reconstructed in the event of container information corruption.

2.3 Third Year Activity

While there were several minor suggestions made during the third phase, only two design significant requests were received – (1) support a native file system usage/layout capability to enable convenient use of the archive format as a program or application specific native database and (2) support user specified units' features to allow arbitrary explicit units definition. The primary feedback from discussions and presentations was a reiteration of the need for an intrinsically, user extensible metadata capability to accommodate unforeseen future informational storage needs.

Overall, the third phase of the project focused primarily upon reducing the many feature suggestions and the observed archive needs to a viable format requirement specification and identifying the minimum required feature set for successful deployment. Most of the rest of the feature requests will be adopted as optional archive extensions.

3. Current Requirements for Specification

This section presents the overarching picture of the format requirement specification as it currently stands and represents the starting point for a potential two or three phase implementation and validation effort.

3.1 Design Challenge: Simplicity vs. Complexity

It is important at this point to recognize the fact that despite the apparent complexity of the proposed solution, the basic simplicity of the UFF structure has been preserved. The apparent complexity of some of the features like the matrix definition, specifically the sub-matrix partitioning scheme, are included to support the needs of certain vendor/user communities. It must be emphasized, however, that it is not necessary to utilize all the features in order to write a compliant archive.

As examples of these differing user community needs, consider the widely variant requirements of the following user scenarios and note the challenge for how the specification addresses each of their unique data archival needs. There are those users with minimal, basic needs (e.g. 4 channel trouble shooting); those with large channel count FRF needs (e.g. 3 in x 250 out); those with high-speed, long record time capture involving multiple test conditions (e.g. jet engine testing); and those acquiring specialized information who require secure, multi-path delivery (military); et al.

The following sections expand upon the various feature suggestions and start to clarify and distinguish between the minimum necessary information and the recommended, but optional, documentary information, thus illustrating the underlying simplicity of the fundamental solution.

3.2 Archive Feature Specifications

During the various formal and informal discussions with users and vendors that have occurred during this project, many suggestions for desirable features were offered which, while perhaps not immediately applicable to the project effort then underway, were worth noting for consideration during future work. Many of the suggestions do not affect the principal data per se, but rather focus on the retention of historical metadata information and the like. Examples of these suggestions and concerns are:

- It should be possible to write verbose output (i.e. redundant info) with equivalence constraint testing capability (e.g. writing multiple measurement vectors from measurement matrix and checking measurement characteristics or constraints. [fmin, deltaf, testid, block length, etc.])
- When preserving data it should be possible to write a verbose output with some form of back trace to the original database fields. (e.g. perhaps writing <meas vendorSource="hatchTest[1]">... data ...</meas> where "hatchTest[1]" may be the original vendor data ID.)
- It might be advantageous to reserve all 'vendorXXX' attribute fields for vendor use.
- It might also be advantageous to reserve all 'userXXX' attribute fields for end-user use.
- In developing the XML data specification, attributes should not provide any data information, but only metadata information about the data.
- It should be possible to tag or log any hardware or software that has touched/modified the data (i.e. retain the data history path.)
- It should be possible to document vendor specific or proprietary information within the container using human readable ASCII/XML - *NOT* PDF/DOC/etc.
- The 'Open Container' should allow inclusion of other non-format defined information types. (e.g. images, sounds, etc.)
- The format should have clearly defined behavior as well as content. (i.e. specified error handling in the presence of malformed data.)
- Inline data should be written in decimal: floating point or bytes. Complex data should be specified as successive pairs of real values.

Although additional feedback was (and is) expected as the project continues to progress, these types of comments favor the development of an 'XMLized' UFF-like format definition. Additionally, many of these suggestions are inherently supported by the working concept through the synergy of the ECMA 'Open Container' (or a close equivalent) coupled with a predominantly XML data definition. Further, an 'XMLized' UFF has the strength of familiarity, thus facilitating community acceptance.

The review of the ECMA 'Open Container' contributed much to the conceptual design of the format, even though strictly the specification will not be used as the primary data container. Strict conformance to the 'Open Packaging' specification has been abandoned due to the risk of single point concentration failure of the record association hash table, as well as, the documented ability to silently replace records. (While the ability may be advantageous for replacing logos and other local document customizations in the field of desktop publishing, the feature represents a significant inadvertent corruption risk.) Since the protection against this (and other) failure requires that complete association information to be stored integrally with each record, there remains no advantage to maintaining this redundancy. (Although performance was not a primary concern for this project, limited testing has indicated that write performance degradation grows with increasing number of archive elements potentially making the archive non-manipulative when size exceeds in memory capacity.) The advantages obtained by abandoning strict conformance also include the ability to support a native file system container basis and the potential to support other (current and future) file container archive formats.

3.3 Feature Elements Driven by Recoverability

Since the overarching principal is long-term recoverability, many of the archive feature characteristics chosen have been governed by that objective. As mentioned before, the only historically successful, long term archival format has been printed matter. Books, papyri, engravings, etc. all yield valuable (and recoverable) information, even when significantly damaged. The following discussion presents the design impact of recoverability upon some of the archive features.

All data shall be written in UTF-8 encoding to facilitate recovery. Because UTF-8 encoding is backward compatible with ASCII, it guarantees that no low order (0x00-0x7F) ASCII characters occur in any multi-byte encoding, thus the data stream is also self-synchronizing. This behavior, coupled with additional constraints, such as requiring all UFR format master control field names to be strict ASCII UPPERCASE (e.g. DRT, VER, LREF, XREF, etc.) and requiring all record specific informational field names to be ASCII MixedCase. (e.g. DataType, TemperatureOffset, Length, etc.), enables more robust data recovery in the event of inadvertent archive corruption.

To facilitate recovery, large data records should be broken into smaller, more manageable pieces (e.g. segments of 50-100 kb.) The various pieces shall be associated using connection references and segment variable features (such

as Fmin or Tmin) shall be adjusted to be correct for each segment. For example, the format (structural arrangement) of time series (function) data must have ability to be partitioned throughout the data stream as needed for best resilience against data corruption. The series shall be broken into a set of manageable pieces, each with individual checksum coding. The checksum encoding must be distinct from the validation code stored in the ZIP container element header. The series pieces can be organized by any of the following from single complete channel record to multiple channels interleaved (with a granularity [blocksize] from complete record down to single point). In order to support this capability properly, it requires that the functional information must intrinsically support multidimensional data.

The archive must contain redundant structural (data organizational) information (preserved with each data record element) in an extractable ASCII readable form. The archive must support redundant (duplicate) data records for key informational content. Also as part of the semantic (informational) structure, each data matrix should receive a unique identification (UID/name) thus also helping to support multiple sets of similar information within the archive and allowing more convenient mapping of vendor database structures.

All field definition (content) strings should be trimmed of leading and trailing white-space. This helps address the issue of the user adding white-space for visual and/or readability purposes, but which is not relevant (or influential) to the information being stored. Thus the ability of the software to read and interpret the informational field correctly is not compromised by a user preference or idiosyncrasy.

Other feature concepts which support recoverability include:

- Each container (ZIP) file entry contains a single archive data record.
- Binary data must NOT be mixed (interspersed) with ASCII (textual) data.
- All basic record field names shall be defined using mixed-case English.
- All archive entry names must be archive root relative.
- All external names must be either archive root relative or file system absolute.
- Each EXT (extension) reference, regardless of being internal or external, must consist of the Path, Name, UID, and Type.

Many of these features have been so chosen in order to facilitate the development and utility of recovery codes capable of scanning a damaged archive and then extracting and reconstructing as much as practical of the original information.

3.4 Format Design Principles

3.4.1 Error Detection / Correction

Because of the block oriented nature of most data storage devices (disks, CDs, DVDs, flash memory, etc.) bit stream encoding errors are unlikely. Generally, the failure will be entire blocks of lost information. Also, since the fundamental block size of the different devices vary (historically discs were 512 bytes, more recent formats are 4096; CD/DVDs are 2048, etc.) and these devices typically use some form of block based bit stream encoding for error recovery purposes anyway, block oriented schemes which preserve the data information more naturally coupled with simpler limited sized (~100k) record based checksums should be preferable.

Therefore, since the process of error detection is focused upon the identification of inadvertent data corruption, not deliberate manipulation, and the proposed record granularity is recommended to be 50-100k, simpler error detection schemes should be adequate. Since an additional goal is to be able to extract readable information from potentially corrupted archives, simpler hashing/message digest type schemes (CRC32, MD5, et al.) should be preferable to encoding schemes (RS, et al.) because the data remains ASCII. For error detection purposes, each field may have an individual hash coding (CRC, et al.) attribute representing the field informational content and shall be processed based upon the native UTF-8 encoding prior to the encoding of any forbidden XML character sequences. (e.g. <GenericInfoField CRC32="A23CDE87">Potentially meaningful informational content</GenericInfoField>)

Just as error correction is outside the scope of this project and should be handled external to the format definition by the media storage system, the security measure of data encryption is also beyond the scope of this project. Technology is expected to continue to grow over the life of the archive and more sophisticated means develop.

Furthermore, by its design, encryption is intended to make extraction of information inherently difficult and is therefore at odds with the goal of long term viability and recoverability.

3.4.2 Data Homogeneity

All data records shall represent conceptually homogeneous information. For example, heterogeneous array information shall be partitioned into homogeneous informational sections and these sections written out individually. As another example, the heterogeneous DSP parameters describing a multi-rate acquisition shall be written as separated DSP records for each rate condition.

3.4.3 Matrix Data Storage Organization

The format of time series and other functional data shall have ability to be partitioned throughout the data stream as needed for best resilience against data corruption. The series shall be broken into a set of manageable pieces, each with individual checksum coding. The encoding shall be distinct from the validation code stored in the ZIP container element header. The series pieces will be arranged by any of the following: single complete channel record, multiple channels interleaved (granularity [blocksize] from complete record down to single point). The pieces should be associated using connection references and variable features (e.g. Fmin) shall be adjusted to be correct for each segment.

Every non-scalar informational element shall be defined as a matrix. This generalized vector/matrix/array data layout shall intrinsically support multidimensional data, as well as, sparse data. The resulting matrix type shall handle the sequencing, sparseness, interleave, storage characteristics, as well as, the data type (integer/float, ASCII/binary) characteristics. The partitioning aspect of the data stream means that sparse data (at least on the macro scale) becomes intrinsically supported.

Thus, write out all matrices (arrays) as inherently sparse. Prefix (attribute) each partition/segment with the starting position, dimensional strides, and dimensional run lengths. Each segment shall contain only a single data type (e.g. real, float, integer, complex, string, etc.) For matrices with heterogeneous elements, write out each distinct type as a separate partition/segment. (e.g. given [1 2 X ; 3 4 Y ; 5 6 Z] – write out as [1 2 ; 3 4 ; 5 6] & [X ; Y ; Z]) For encoding purposes, all format based subscripts are (1-based) natural numbers (1, 2, 3, etc.) Also, identify indexing permutations (e.g. [1 2 3], [2 3 1], etc.), as well as, stride/blocking/run length structure (e.g. [1:10 2:15 3:1024], dim:len) for partitioned storage. With the chunk size limited to about 100k, a coarser explicitly defined blocking scheme is probably unwarranted as the pieces can be scattered purposefully about the archive to achieve the same result.

When writing out any generalized matrix, the values of the dimensional axes (e.g. frequency, temperature, DOF, etc.) shall be recorded for each partition. By doing so, each partition segment is individually structurally complete. When writing out any generalized (multi-dimensional) matrix, extra-dimensional information may be written either explicitly as additional dimensions or implicitly as coordinated metadata. For example, a single channel waterfall plot of response amplitude vs. both frequency and speed could be written explicitly as a 2D matrix with dimensional axes of frequency (Hz) and speed (RPM). Alternatively, it could be written as a series of 1D matrices with a dimensional axis of frequency (Hz) and a coordinated metadata entry of speed (RPM). This facility for writing matrix information using explicit or implicit axis information shall be fully scalable to any number of dimensions; the limiting (and perhaps absurd) case being the writing of a single scalar value and all dimensional axis information in metadata form.

For measurement data (time/frequency/etc. or FRF/COH/et al.), the matrix coordinate encoding and dimensionality can include: channels (input, output), temporal (time, frequency), auxiliary axis (temperature, speed, altitude) although generally this axis (if scalar) should be stored as metadata. In general, the dimensionality could then be encoded in a natural form, that is, an FRF could be written as 3D even when input-output dimensions are singleton DOFs (and by analogy, MCOH as 2D, etc.).

3.4.4 Information Encoding

Because of the overarching issue of recoverability, the importance of the numeric, vector/matrix data representation cannot be overstressed. Specifically, representing data as ASCII numeric should be the primary practice for long term archiving, since the primary objective is long term recoverability. However, for short term storage and transport, binary encoding (for size) may be more appropriate. Finally, for temporary or transient intermediate

usage, a strict native binary representation may be most appropriate. The key decision point is the cost of error recovery (e.g. re-write the file, re-take the test, degrade the analysis, irreplaceable or total loss, etc.) In many respects this is also affects the issues of redundancy. Hence the need for redundant (backup) records for key failure point records (e.g. units)

All record entries shall use verbose textual field enumerations instead of numerical coding. Although verbose field identifiers and content admit the possibility of spelling errors that do not occur with numeric field enumerations (e.g. 1=FRF, 2=COH, etc.) the advantages outweigh the potential inconveniences, since misspelled words are often easy to correlate with their properly spelled counterpart, recovery from such errors is more easily accomplished. Recovery in the case of actual miscoding is more challenging as there is no redundant information. Of course, true coding errors such as labeling an FRF as a COH is not addressed by either scheme and such detection involves more sophisticated data analysis and is always problematic.

To prevent an artificial inflation of apparent informational precision, there should be a mechanism for indicating the numerical data precision, particularly of floating point data, both for the original information and for the information as encoded in the archive.

3.4.5 Informational Disambiguation

To support the long term viability goal, each data record shall contain a record versioning string (e.g. X.Y.Z, where X = Major Revision - Addition of new fields; Y = Minor Revision - Addition of new field values; Z = Patch - Correction of spelling or clarification of decoding) defining either the minimum specification under which the record can be decoded or the current record specification at the time of writing.

To help support multiple sets of similar information within the archive each data matrix shall receive a unique identification (UID/name).

When redundant records are used one record shall be designated as the master record and all other redundant records as slaves. However, in the event of corruption of the master record, the first valid redundant reference shall become master.

Because of the potential for misinterpretation for a ‘successive pairs of real values’ encoding of complex data, particularly in the event of data recovery, the information should be disambiguated by requiring an explicit complex format (e.g. 1.2+3.4i, 5.6e+7-8.9e0j, -1.4-j6.2, etc.)

To reduce the potential for misinterpretation, an explicit units suffix capability has been suggested (e.g. <Frequency>1.35 Hz<Frequency/>, where ‘Hz’ is defined in a units system suffix table.)

Within the dynamics test community, there are numerous nomenclature terminologies which can often refer to the same or similar informational content. For example:

- Ensembles, Scans, Traces, Functions, Maps, Blocks, ...
- BlockSize, Span, Number of Spectral Lines, ...
- Projects, Groups, Sessions, ...
- Elements, Records, DataSets, ...

Such common cross terminology usages shall be footnoted for each record or informational field type.

All field definition (content) strings shall be trimmed of leading and trailing white-space. This helps address the issue of the user adding white-space for visual and/or readability purposes, but which is not relevant (or influential) to the information being stored. Thus the ability of the software to read and interpret the informational field correctly is not compromised by a user preference or idiosyncrasy.

3.4.6 Record/Field Detail Notes

The field names and types are designed to map naturally to various programming language integral types like structures, cell arrays, and/or name value pairs.

All UFR format master control fields shall be strict ASCII UPPERCASE. (e.g. DRT, VER, LREF, XREF, etc.)
All record specific informational fields shall be ASCII MixedCase. (e.g. DataType, TemperatureOffset, Length, etc.)

An empty field (e.g. <EmptyFieldName/>) shall be considered a null value for numeric fields and an empty string for character values.

Each time an archive is touched (modified), a comment record should be added to the MASTER header (EXT). Each EXT reference, regardless of being internal or external, shall consist of the target name, UID, Version, and Type.

Where data fields contain coordinated data, it is the responsibility of the generating code to maintain consistency and it is the responsibility of the consuming code to detect inconsistency.

All data shall be written in UTF-8 encoding to facilitate recovery and must include the XML prologue material - <?xml version="1.0" encoding="UTF-8" ?>

4. Summary / Conclusions

This paper documents the first three years of effort and leaves the design in a form that can be resumed in any potential follow on phase of work.

Over the course of the project, numerous formal and informal discussions were held. These discussions were intended to represent a cross-cutting industry comment solicitation activity of users, vendors, and government installations. Overwhelmingly, those contacted expressed support for the project and many times the responses could be summarized as ‘sounds good’, ‘this is needed’, or ‘let us know when it’s ready’. All significant design comments and suggestions have been evaluated and integrated as appropriate and even those comments which were not specifically immediate design relevant were preserved against potential relevance in future phases.

The result has been that most of the requirements for that absolute minimum set of features which a vendor/user must use in a compliant archive have been identified. Additional features that a vendor has the option to use have also been identified. Some features, however, only make sense to completely define during an implementation stage as they can affect recoverability and/or performance.

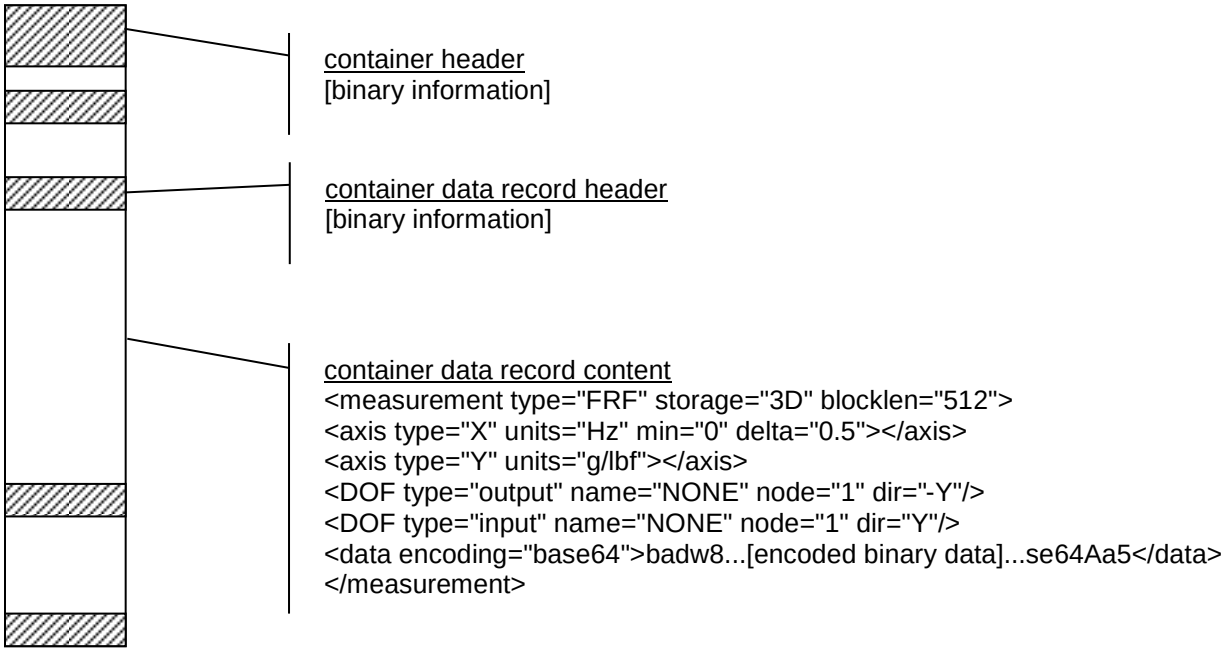
Bibliography

- Phillips, A.W., Allemang, R.J., “Requirements for a Long-term Viable, Archive Data Format”, IMAC, 2010, 5 pp.
- Phillips, A.W., Allemang, R.J., “Archive Format Requirements for Long Term Storage of Dynamic Signal Data”, ISMA, 2010, 8 pp.

Appendix A

A.1 Pictorial Concept Example

The following example is presented for conceptual discussion purposes, giving only an impression of the style of data storage. It is not intended to be complete or to represent any particular likely final implementation.



Note that, in support of the different envisioned operational usage paradigms, additional data encoding/representations are planned, specifically an ASCII decimal and a referenced native binary.

Appendix B

The following example data records are not intended to represent a complete definition. Instead, they are presented in order to provide a sense of the initial expected representation for a few of the principal required data records. Numerous other key records, not listed here, are also in draft form.

B.1 Master Data Record

Field Name	Count	Content	Comments
DRT	1	String	Data record type or kind
VER	1	String	Record format revision
UID	1	String	Unique record identification string
SELF	1	String	Self-referential archive root relative path
EXT	0-1	String Array	List of extension record references

Table 1: Prototypical Master Field Definition – Information common to all records

Record Notes:

- Whitespace encoding needs to differentiate between format-free and user-defined whitespace information. ‘blanks’, ‘tabs’, ‘newlines’, ‘carriage returns’, ‘line feeds’, etc. are consider format-free, that it, they can be changed, replaced, or expanded at will or need without regard to usage. Such whitespace is useful for field value separation and delimiting.
- If user-defined whitespace is to be preserved without modification, it shall be explicitly encoded (%20, %09, %0A, etc.) within the record.
- For error detection code (CRC32, et al.) purposes, all contiguous format-free whitespace shall be treated as a single ASCII blank character (0x20).

B.2 Reference Record

Field Name	Count	Content	Comments
Target	1	String	Target record - pathname
TargetUID	1	String	Target record UID
TargetVER	1	String	Target record VER
TargetDRT	1	String	Target record DRT

Table 2: Prototypical Master Reference Definition

Record Notes:

- The self/cross reference record structure capability and behavior. In particular how external archive references should be able to tunnel into other archives. The potential usages of this feature include: scrubbed data reassembly (i.e. from delivery via different paths), correlated test info, etc. While such a feature would be a poor choice to use for long term storage, it may be imperative for secure delivery or transport. (And hence impact certain industry acceptance.)
- Particularly for the support of "external/scrubbed" data content, the concept possibility implementing inline references through attributes (e.g. <DataContent Target="/Measurement/Units/UnitSet1.DSR" TargetUID="Units-1a" TargetVER="1.0.0" TargetDRT="Units" />) should be evaluated. This could also become the mechanism basis for referencing native binary content (e.g. either by referencing the binary record or direct reference coding.)

Concept of References - local (within archive), external (to file system or other archive)

aref - absolute reference
lref - local reference
rref - relative reference
xref - cross reference (or maybe back/return reference)

Example: Generic data driven reference record

<REF>
<Target> /Measurement/Units/UnitSet1.DSR </Target>
<TargetUID> Units-1a </TargetUID>
<TargetVER> 1.0.0 </TargetVER>
<TargetDRT> Units </TargetDRT>
</REF>

Example: Generic attribute driven reference record

<REF Target="/Measurement/Units/UnitSet1.DSR" TargetUID="Units-1a" TargetVER="1.0.0" TargetDRT="Units" />

The final decision on whether or not REF(erences) should be "data" driven or "attribute" driven, along with the relative advantages and disadvantages of each approach, will be resolved during the initial reference implementation phase.

B.3 File Header Record

Field Name	Count	Content	Comments
DRT	1	String	Data record type or kind
VER	1	String	Record format revision
UID	1	String	Unique record identification string
SELF	1	String	Self-referential archive root relative path
EXT	0-1	String Array	List of extension record references
FileStamp	1	String	
DateStamp	1	Date String	
TimeStamp	1	Time String	
CreationDate	1	Date String	
CreationTime	1	Time String	
DatabaseVersion	1	String	

Table 3: File Header Definition

File Header Example

<UFR>
<DRT>FileHeader</DRT><VER>1.0.0</VER><UID>FileHeader-1a</UID>
<SELF>/FileHeader1.DSR</SELF><EXT/>
<FileStamp> C:\Projects\PlateTest\Test1.DSA </FileStamp>
<DateStamp> 14-July-2010</DateStamp><TimeStamp> 14:23:05 </TimeStamp>
</UFR>

B.4 Comment Record

Field Name	Count	Content	Comments
DRT	1	String	Data record type or kind
VER	1	String	Record format revision
UID	1	String	Unique record identification string
SELF	1	String	Self-referential archive root relative path
EXT	0-1	String Array	List of extension record references
Description	1	String	User meaningful description
CommentLog	0+	String	Sets of user comments/logs

Table 4: Comment Record Definition

Comment Example

<UFR>
<DRT>Comment</DRT><VER>1.0.0</VER><UID>Comment-1a</UID>
<SELF>/CommentRecord1.DSR</SELF><EXT/>
<Description>First principal c-plate test.</Description>
<CommentLog>3-August-2010 16:06 - Initial dispersion failed. Will reattempt using second generation mark 2!</CommentLog>
<CommentLog>5-August-2010 08:43 – Test cancelled. Second delivery aborted.</CommentLog>
</UFR>

B.5 Units Record

The Units Record shall be used to define the system of units employed, not the specific units of any particular data element.

Field Name	Count	Content	Comments
DRT	1	String	Data record type or kind
VER	1	String	Record format revision
UID	1	String	Unique record identification string
SELF	1	String	Self-referential archive root relative path
EXT	0-1	String Array	List of extension record references
System	1	String	Canonical units system name
Length	1	Scalar Float	Conversion factor to SI unit (meter)
Force	1	Scalar Float	Conversion factor to SI unit (Newton)
Temperature	1	Scalar Float	Conversion factor to SI unit (K)
TemperatureOffset	1	Scalar Float	Temperature base reference
TemperatureMode	1	String	Temperature measurement mode (abs/rel)
Time	1	Scalar Float	Conversion factor to SI unit (sec)
Current	1	Scalar Float	
LumenalIntensity	1	Scalar Float	
Ohms	1	Scalar Float	
Mole	1	Scalar Float	
PlaneAngle	1	Scalar Float	

Table 5: Units Record Definition

Record Notes:

- The Units Record descriptor will need to expand to accommodate the suggested 'UnitSuffixTable' descriptor.
- When using the 'UnitsSuffixTable', all unit encodings should utilize as near as possible to the standard units names or abbreviations. For example, 'Hz' shall refer to a frequency unit of '1/sec'.
- The use of the 'UnitsSuffixTable' for informational obfuscation should be limited to specialized security needs.

Units Example

```
<UFR>
<DRT>Units</DRT><VER>1.0.0</VER><UID>Units-1a</UID>
<SELF>/Measurement/Units/UnitSet1.DSR</SELF><EXT/><System>SI</System>
<Length>1.0</Length><Force>1.0</Force><Temperature>1.0</Temperature>
<TemperatureOffset>273.15</TemperatureOffset><TemperatureMode>absolute</TemperatureMode>
</UFR>
```

Appendix C

C.1 Other Content Information

The definition and development of results records refers primarily to retained computational results. In one sense, this is an almost unending task; however, initially this should be limited to the most common dynamics measurement processed results (e.g. modal parameters [UF55] & general matrices [mass/stiffness/damping/etc.]). Note that this would not be computed measured results like FRF, COH, and the like, which are already included in the dynamic data measurement record.

While not truly dynamic information, but for completeness and successful community adoption, a set of geometric information records corresponding roughly to the UFF records nodal coordinates (UF15), components, coordinate systems (UF18), trace lines (UF82), etc. needs to be developed.

C.2 User Extensibility

Follow on implementation efforts need to include the development of sets of convenient, extensible metadata record definitions for non-critical common data documentary information which include the metadata record name, user defined name-value pair information, and value data type definition (char, string, numeric, integer, floating point, complex, vector, matrix, etc.)

Finally, the implementation of user records needs to clearly define the process of creating user/custom data records which should include the potential for embedded syntactic/semantic user documentation. The primary point of this is to allow user developed records to be self-documenting.