

Evaluating a Format for Viable, Long-Term Dynamic Data Archiving

Allyn W. Phillips and Randall J. Allemang, University of Cincinnati, Cincinnati, Ohio

This article reviews the various (and sometimes conflicting) issues involved in the development of a new flexible, open-definition file format. To help facilitate early evaluation of format function and performance, an intermediate pseudo-prototypical hybrid UFF-DSA (dynamic signal archive) format has been implemented. This evaluation format effectively encodes the UFF dataset record information in a XML format (similar to the draft DSA definition) and then stores the individual XML records as discrete entries in the DSA archive container (ZIP format). In addition, the resulting principal features necessary for viable long term archiving of data and results, together with an evaluation of an initial implementation of the design, are presented.

For many years, the universal file format (UFF) has served as the *de facto* standard for open data interchange. However, as technology has progressed, the aging nature of its 80-character line oriented, ASCII, FORTRAN card, image-based format has become problematic. So there has developed within the vibration technical community a need for a long-term-viable, open-definition file format for the archiving of dynamic signal data and results; one which is independent of any particular hardware or operating system environment and distinct from any particular database management structure.¹⁻³ Following a brief discussion of the project background and history, this article focuses on the feature set identified for realistic, long-term reliable recovery of information and successful future community adoption.

Overall, the project is focused on the longterm archive of dynamic measurements and associated metadata. Performance and size of the archive are not the primary objectives; data integrity and recoverability are the prime objectives. The problems associated with malicious damage are specifically outside the scope, as are the issues of refreshing the data as media storage technology changes. Instead, the project is limited to the detection of inadvertent data corruption and extraction of remaining valid data.

One of the significant challenges to this project has been that there are very few data formats that are both open and usable without restriction. Therefore, the existing UFF was taken as the initial starting reference point.

Background

The first phase activity of the project focused on soliciting

Based on a paper presented at IMAC-XXX, the 30th Conference & Exposition on Structural Dynamics, Jacksonville, FL, January/February 2012.

technical community (industry, government, and academic, both vendors and users) feedback regarding their data archive expectations and needs. Using that feedback, the project continued with the identification and evaluation of existing formats as potential format bases. Most of the formats, besides not meeting all the goals of the project, could not be seriously considered due to legal usage restrictions. However, at the conclusion of the first phase, three potentially viable foundational solution options (or directions) had been identified.

- The first was a DOE sponsored effort, the HDF (hierarchical 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 on the data archive project sponsor needs and the aggregated vendor/user feedback.

During the second phase of activity, the process of reviewing the three primary archival format candidates proceeded, in part, by reviewing existing available data format options with a specific focus on applicable features for incorporating into the resulting 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 were still relevant. Consideration was also given to the feature characteristics needed for long-term read/recovery viability and industry acceptance. As a result, a format that is principally textual (ASCII) encoded data was favored because of its nominal familiarity, its ease of implementation, and its relatively straightforward mapping to existing proprietary databases. So the operating plan at the conclusion of year two was to use a restricted format, industry standard ZIP file for the primary data container, the contents of which were envisioned primarily as sets of XML data streams.

Although format feature suggestions continued to be solicited and evaluated (and a couple design-significant requests were received and adopted), overall, the third-phase activity of the project focused primarily upon reducing the many valuable feature suggestions and the observed archive needs to a viable format requirement specification and identifying the minimum required feature set for successful deployment.

For an extensive discussion of the development history, guiding principles, and technical decisions, see References 4 and 5.

DISCLAIMER

This work of authorship and those incorporated herein were prepared by Contractor as accounts of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Contractor, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, use made, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency or Contractor thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency or Contractor thereof.

COPYRIGHT NOTICE

This document has been authored by a subcontractor of the U.S. Government under contract DE-AC05-00OR-22800. Accordingly, the U.S. Government retains a paid-up, nonexclusive, irrevocable, worldwide license to publish or reproduce the published form of this contribution, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for U. S. Government purposes.

Specification Summary

It is important to recognize that the only goal was to produce a long-term, viable, cross-platform, open-architecture, dynamic, data storage format. Practically speaking, that means the format must allow users to export data into a data archival structure that is independent of a 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). So one overarching principle is recoverability! As a long-term archive format, any feature that jeopardizes recoverability must be carefully evaluated. Also, since the content focus is dynamic data, other data content types, such as CAD/CAE, video, pictures, etc., are not specifically included in the format. But such information can be referenced either via the metadata records or via inclusion within the data archive container.

The principle of “keeping it simple” has been followed to facilitate both industry acceptance and long-term comprehension. The result has been a format that preserves much of the basic simplicity of the UFF structure. The apparent complexity of some of the features has been included to address the needs of certain vendor/user communities. As examples of the differing user community needs and their widely variant requirements, the following user scenarios illustrate the challenge to the specification to address each of their unique archive needs. There are those users with minimal, basic needs (e.g. four-channel trouble shooting); those with large channel count FRF needs (e.g., 3 in × 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, multipath delivery (military); along with others.

Since the overarching principle is long-term recoverability, many of the archive feature characteristics have been chosen with that objective in mind; consequently, several archive format characteristics are fixed. All data are written in UTF-8 encoding; 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 producing a data stream that is also self-synchronizing. All archive names in the master control field are required to be strict ASCII uppercase, and all record-specific informational field names are required to be ASCII mixed-case, enabling more robust data recovery in the event of inadvertent archive corruption. Large data records are broken into smaller, more manageable pieces of approximately 50-100 kb. The archive also contains redundant data organizational information on a record-by-record basis in an extractable ASCII-readable form as well as supporting redundant (duplicate) data records for key informational content.

Implementation Objectives

The principal objective of the fourth phase of the project was to evaluate the DSA draft specification design through implementation within a single programming environment. To facilitate testing, the environment chosen for phase four was MATLAB, because the UC-SDRL (University of Cincinnati – Structural Dynamics Research Lab) research software is implemented in that environment. (Note that should another phase be supported, the objective will be to port the code to additional operating environments.) Overall, the objective is to develop an open set of software libraries that implement all key features of the DSA draft specification but are unencumbered by proprietary licenses. Therefore, all code development has been new.

As design work on the initial reference implementation progressed (within the MATLAB environment), the primary development activities and design decisions focused on the application programming model and its impact on final industry receptivity. The issues associated with the generation and access of the DSA archive container (ZIP format) from MATLAB and the generation and parsing of the DSA archive content record (XML format) information were key. So the work on the low-level access of the DSA archive container (ZIP format) from within the MATLAB environment and the generation and parsing of the DSA archive content record

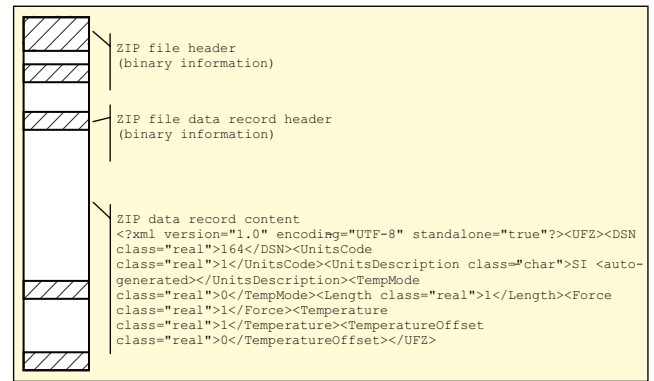


Figure 1. Archive layout structure.

(XML format) information were feature areas of focus.

Universal File Zip

To help facilitate component testing and to allow early evaluation of format performance and function within the existing UC-SDRL research software, an intermediate pseudo-prototypical hybrid UFF-DSA format was implemented. The evaluation format effectively encoded the universal file format (UFF) dataset record information in an XML format (similar to the DSA draft specification) and then stored the individual XML records as discrete entries in the DSA archive (Figure 1), exercising key elements of the DSA specification.

During implementation, it became clear that encoding the data class (char, int, real, complex, etc.) provided great advantages in preserving and decoding the XML field contents. See Examples 1 and 2.

Evaluating the functional behavior demonstrated both the effectiveness of the DSA specification model, as well as the desirability for a relatively simple and independent archive library. Part of the functional behavior evaluated included the ability to both store or compress records (independently) within the universal file zip (UFZ) container. The result was that performance was very reasonable and, because of the preponderance of measurement data (UFF Type 58), the size of the resulting files were comparable to their normal UFF counterparts; that is, the stored UFZ files were overall comparable in size to the ASCII UFF, and the compressed UFZ files were comparable to the binary UFF. Experience gained from the interim UFZ format has greatly influenced the resulting DSA implementation.

Dynamic Signal Archive

Development and testing on the intermediate pseudo-prototypical hybrid UFF-DSA format continued as more features of the full DSA specification were implemented and evaluated. In addition, the effort revealed the need and advantages of developing a compact, independent parser-archiver. Besides simplifying programmatic usage, it was expected to help facilitate the process of porting the library routines in the future.

With the concentration of effort focused on the temporal measurement-based record needs (time and frequency), support for a number of key features has been implemented, including:

- Uncompressed and compressed records within the ZIP container
- Direct read and write of DSA records to and from the MATLAB environment
- Proposed unit suffix table capability
- Auto/programmatic split/segmenting of temporal measurement records and general data matrices
- Backup/redundant records
- Attribute-driven reference records

To help give a sense of the state of this development and its progress, a list of selected comments on two actual records (Examples 3-4) follows:

- Every informational content field has a data “class” attribute indicating the native data characteristic: char, integer, real, complex, struct (Examples 3-4).
- The suffix table field of the unit system record shows how the

```

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<UFZ>
  <DSN class="real">151</DSN>
  <ModelFileName class="char">D:\ Research\X-Modal II\data\cplate_w_geo.ufb</ModelFileName>
  <ModelFileDescription class="char">X-Modal DMGR</ModelFileDescription>
  <ProgramDB class="char">X-Modal DMGR</ProgramDB>
  <DateCreated class="char">04-Jun-2006</DateCreated>
  <TimeCreated class="char">10:41:26</TimeCreated>
  <DateSaved class="char">04-Jun-2006</DateSaved>
  <TimeSaved class="char">10:41:56</TimeSaved>
  <ProgramUF class="char">X-Modal DMGR</ProgramUF>
  <DateWritten class="char">04-Jun-2006</DateWritten>
  <TimeWritten class="char">10:42:25</TimeWritten>
</UFZ>

```

Example 1. Universal file header record (Type 151).

```

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<UFZ>
  <DSN class="real">82</DSN>
  <TraceLine class="real">1</TraceLine>
  <TraceLength class="real">47</TraceLength>
  <Color class="real">1</Color>
  <ID class="char">User Component</ID>
  <Trace class="real">1 13 25 0 2 14 26 0 3 15 27 0 4 16 28</Trace>
</UFZ>

```

Example 2. Universal file trace record (Type 82).

```

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<DSR>
  <DRT class="char">UnitSystem</DRT>
  <VER class="char">0.0.1</VER>
  <UID class="char">UNITS-1a</UID>
  <SELF class="char">dsr-units</SELF>
  <EXT/>
  <System class="char">SI</System>
  <Factors>
    <Mass class="real">1</Mass>
    <Length class="real">1</Length>
    <Time class="real">1</Time>
    <Current class="real">1</Current>
    <Temperature class="real">1</Temperature>
    <TemperatureOffset class="real">0</TemperatureOffset>
    <LuminalIntensity class="real">1</LuminalIntensity>
    <Mole class="real">1</Mole>
    <PlaneAngle class="real">1</PlaneAngle>
  </Factors>
  <SuffixTable class="struct" index="1">
    <Label class="char">kg</Label>
    <Scale class="real">1</Scale>
    <MassExp class="real">1</MassExp>
    <LengthExp class="real">0</LengthExp>
    <TimeExp class="real">0</TimeExp>
    <CurrentExp class="real">0</CurrentExp>
    <TemperatureExp class="real">0</TemperatureExp>
    <LuminalIntensityExp class="real">0</LuminalIntensityExp>
    <MoleExp class="real">0</MoleExp>
    <PlaneAngleExp class="real">0</PlaneAngleExp>
  </SuffixTable>
  <SuffixTable class="struct" index="2">
    <Label class="char">Hz</Label>
    <Scale class="real">6.283185307179586</Scale>
    <MassExp class="real">0</MassExp>
    <LengthExp class="real">0</LengthExp>
    <TimeExp class="real">-1</TimeExp>
    <CurrentExp class="real">0</CurrentExp>
    <TemperatureExp class="real">0</TemperatureExp>
    <LuminalIntensityExp class="real">0</LuminalIntensityExp>
    <MoleExp class="real">0</MoleExp>
    <PlaneAngleExp class="real">1</PlaneAngleExp>
  </SuffixTable>
</DSR>

```

Example 3. Dynamic signal archive, units system record.

- index (position) of structured arrays is preserved. (Example 3).
- The SELF field of the matrix record shows how the split/segmented and backup/redundant records are implemented. In this case, the record is the first backup record of the sixth piece of a total of 43 parts. (Example 4).
- The unit system field of the matrix record shows the reference mechanism to the unit system record (Example 4).

- The matrix.partition.data field of the matrix record shows how data type, dimensioning, and element indexing is preserved and recorded (Example 4).
- The units field of the matrix records shows the connection by name to the suffix table defined in the associated unit system field, which in this case references a separate units system record (Example 4).

Although not shown in the examples, support for arrays containing heterogeneous informational content has also been implemented. Note that many of these implemented features also support the goal of producing a format that is naturally extensible in the future and provide information that makes the programmatic connection between the DSA archive and the native system/environment structure both simpler and more robust.

Import Convertors

In addition to the archive library routines, a secondary objective is to produce a set of import convertors for different common file types to facilitate the translation and integration of archive informational content. Presently support exists for conversion of universal file format (UFF) to/from dynamic signal archive (DSA) and standard data format (SDF) to dynamic signal archive (DSA). Currently, the UFF/DSA convertor focuses on measurement data (UFF type 58 and related records) and the SDF/DSA convertor focuses upon

long-time record data throughput. Other convertors are envisioned as the project proceeds.

Summary/Conclusions

This paper documents the interim state of the fourth phase effort and represents an implementation of the previously identified absolute minimum set of features that a vendor/user must use in a

```

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<DSR>
  <DRT class="char">Matrix</DRT>
  <VER class="char">0.0.1</VER>
  <UID class="char">MTX-1a.part[6#43].bkp[1]</UID>
  <SELF class="char">dsr-mtx.part[6#43].bkp[1]</SELF>
  <EXT/>
  <Description class="char">General Matrix</Description>
  <UnitSystem>
    <REF TargetUID="UNITS-1a" TargetVER="0.0.1" TargetDRT="UnitSystem" Target="dsr-units"/>
  </UnitSystem>
  <Units class="char">kg</Units>
  <Matrix>
    <Dimension class="integer" dimension="1 3" indexing="1 2">7 36 512</Dimension>
    <Partition>
      <Base class="integer" dimension="1 3" indexing="1 2">1 1 61</Base>
      <Stride class="integer" dimension="1 3" indexing="1 2">1 1 1</Stride>
      <Data class="complex" dimension="7 36 12" indexing="1 2 3">1.95112e-007 3.21146e-008
        -1.87252e-007 1.03019e-008 -1.82404e-007 3.02997e-009 -3.81776e-008 1.69678e-008
        3.63597e-008 3.63597e-009 9.33231e-008 7.87793e-009 1.69517e-008 0 -3.93896e-008
        1.21199e-008 5.45343e-008 -8.48311e-009 -3.0115e-007 9.69498e-009 1.86022e-007
          [*** numeric data elided for presentation purposes ***]
        1.80464e-007 -5.62597e-008 -9.30272e-009 2.22833e-008 -5.84236e-008 2.3153e-008
        -7.78981e-009 9.52088e-009 1.17929e-008 -7.898e-009 -1.03844e-008 3.41821e-008
        -1.16836e-007 5.40907e-008 7.85472e-008 -2.05564e-008 -6.96622e-008 5.4951e-008
        1.76353e-008 6.92428e-009 4.04637e-008 1.30912e-008 4.21948e-009</Data>
    </Partition>
  </Matrix>
</DSR>


```

Example 4. Dynamic signal archive, matrix record.

compliant archive. Testing and evaluating initial reference implementation suggests the expected success for the overall project. The first public release of a MATLAB interface is expected to be completed and available in early 2013, with other programming language interfaces expected to follow later in the year.

References

1. Phillips, A. W., Allemang, R. J., "Requirements for a Long-Term Viable, Archive Data Format," IMAC, 2010.

2. Phillips, A. W., Allemang, R. J., "Archive Format Requirements for Long-Term Storage of Dynamic Signal Data," ISMA, 2010.
3. Allemang, R. J., "Back Up Your Data – The Need for a New Data Archival Format," *Sound & Vibration*, Vol. 45/No. 1, 2011.
4. Phillips, A. W., Allemang, R. J., "Development of a Long-Term Viable Data Archive Format," IMAC, 2011.
5. Phillips, A. W., Allemang, R. J., "Evaluation of a Long-Term Viable Dynamic Data Archive Format," IMAC, 2012. 

Written comments or questions via e-mail are welcome: allyn.phillips@uc.edu.