Low-Level Data Containers

Morgan C White

Los Alamos National Laboratory

Submitted to the May 2013 Meeting of the WPEC Subgroup 38

LA-UR-13-X

Abstract

The successor to the ENDF-102 data format has been discussed many times. The NEA WPEC SG38 has been established to make the latest attempt. This set of viewgraphs contains some thoughts on low-level primitives used in ENDF, in ways that exemplify the good, the bad and the ugly. Hopefully, lessons learned from 50 years of experience with these will guide decisions made in the current proposal. Some thoughts will also be offered on the specific requirements for new primitives in the hope they will generate discussion and debate during the upcoming meeting with the goal of generating a time line of tasks and tasking for delivery of the requirements, documentation and coding for low-level primitives to be used by a new nuclear data format.

Great Works Must Be Built On Solid Foundations

- **The low-level data containers are the foundation upon which everything else is built**
- **Development plans**
  - Today
    - Review existing containers
    - Discuss what is needed & how to do it
  - Next six months
    - Develop requirements & prototypes
  - November 2013
    - Review initial draft
  - Next twelve months
    - Tweak as needed
  - November 2014
    - Finalize documentation and coding

ENDF
The Good

- …
- **Build high-level structures using simple primitives**
  - Keep low-level primitives to a minimum
  - Make NO exceptions (well, almost)
- …

What a beautiful thing! One can write five low-level read/write routines and read ALL of ENDF.

ENDF-102
Five Basic Records

- **TEXT Record**
  - [MAT, MF, MT/ HL] TEXT
  - READ(LIB,10)HL,MAT,MF,MT,NS
  - 10 FORMAT(A66,I4,I2,I3,I5)

- **CONT Record**
  - [MAT,MF,MT/C1,C2,L1,L2,N1,N2]CONT
  - READ(LIB,10)C1,C2,L1,L2,N1,N2,MAT,MF,MT,NS
  - 10 FORMAT(2E11.0,4I11,I4,I2,I3,I5)

- **LIST Record**
  - [MAT,MF,MT/ C1, C2, L1, L2, NPL, N2/ Bn] LIST
  - READ(LIB,10)C1,C2,L1,L2,NPL,N2,MAT,MF,MT,NS
  - 10 FORMAT(2E11.0,4I11,I4,I2,I3,I5)
  - READ(LIB,20)(B(N),N=1,NPL)
  - 20 FORMAT(6E11.0)

- **TAB1 Record**
  - [MAT,MF,MT/ C1, C2, L1, L2, NR, NP/xint/y(x)]TAB1
  - READ(LIB,10)C1,C2,L1,L2,NR,NP,MAT,MF,MT,NS
  - 10 FORMAT(2E11.0,4I11,I4,I2,I3,I5)
  - READ(LIB,20)(NBT(N),INT(N),N=1,NR)
  - 20 FORMAT(6I11)
  - READ(LIB,30)(X(N),Y(N),N=1,NP)
  - 30 FORMAT(6E11.0)

- **TAB2 Record**
  - [MAT,MF,MT/ C1, C2, L1, L2, NR, NZ/ Zint]TAB2
  - READ(LIB,10)C1,C2,L1,L2,NR,NP,MAT,MF,MT,NS
  - 10 FORMAT(2E11.0,4I11,I4,I2,I3,I5)
  - READ(LIB,20)(NBT(N),INT(N),N=1,NR)
  - 20 FORMAT(6I11)

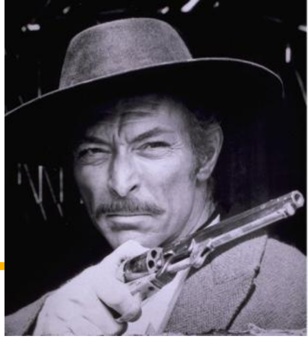
Hold your horses, we'll get to the others

How To
Read ALL of ENDF

- **Loop Until End-Of-File**
  - Sequentially read and store each individual record
  - Read each record, catching exception and trying next type, as …
    - TAB1, TAB2, LIST, CONT, or TEXT
    - Reading in this order will find most complicated record
      - Though there is a small but non-negligible chance of getting it wrong
      - Most generally this is multiple CONT records recognized as a LIST

- **Result is an array of ENDF records**
  - This array can be tested for conformity to the ENDF procedural rules
  - Or, it can be queried for data based on external knowledge

ENDF
The Bad



- …
- **Access all data sequentially**
  - Line numbers (really? in the 21st century?)
  - MAT/MF/MT do not equate to self-describing data
  - Define the meaning of all records based on its position in the sequence
- …

This is the fundamental historic *feature* carried over from punch cards and tape access and is the *most painful* limitation in ENDF-102.

ENDF, The Bad
Five Basic Records Except…

- **HEAD Record**
  - It's just a CONT record with a fancy name
  - Worse, it's a procedural requirement masquerading as a record
    - That is, all sections must start with a CONT record
    - And that record must have C1 equal ZA and C2 equal AWR
- **END Record**
  - Again, it's just a CONT record with a fancy name
  - Again, worse it is procedural requirements regarding section labeling
  - Oh, EXCEPT it allows one to leave all the values blank
    - Which Fortran interprets just fine as all zeros
- **DIR Record**
  - Again, it's just a CONT record with a fancy name
  - Oh, EXCEPT it allows one to leave all the values blank
    - Which Fortran interprets just fine as all zeros

ENDF
The Ugly

- …
- **Make NO Exceptions**
  - If only it were so…
- …

Why Forcing Things Is Always A Bad Idea

- INTG records
  - [MAT, MF, MT / II, JJ, KIJ] INTG
  - PARAMETER (NROW=18)
  - DIMENSION KIJ(NROW)
  - IF(NDIGIT.EQ.2) READ(LIB,20) II,JJ,(KIJ(K),K=1,18),MAT,MF,MT,NS
  - IF(NDIGIT.EQ.3) READ(LIB,30) II,JJ,(KIJ(K),K=1,13),MAT,MF,MT,NS
  - IF(NDIGIT.EQ.4) READ(LIB,40) II,JJ,(KIJ(K),K=1,11),MAT,MF,MT,NS
  - IF(NDIGIT.EQ.5) READ(LIB,50) II,JJ,(KIJ(K),K=1, 9),MAT,MF,MT,NS
  - IF(NDIGIT.EQ.6) READ(LIB,60) II,JJ,(KIJ(K),K=1, 8),MAT,MF,MT,NS
  - 20 FORMAT (2I5,1X,18I3,1X,I4,I2,I3,I5)
  - 30 FORMAT (2I5,1X,13I4,3X,I4,I2,I3,I5)
  - 40 FORMAT (2I5,1X,11I5, I4,I2,I3,I5)
  - 50 FORMAT (2I5,1X, 9I6,1X,I4,I2,I3,I5)
  - 60 FORMAT (2I5, 8I7, I4,I2,I3,I5)

Is this truly better than a list?

Formats matter.

Don’t screw it up.

(Pardon for the Americanisms.)

So, What To Do?

KISS – Keep It Simple Stupid

- **At it's most basic, everything is simply a vector of numbers**
  - Think about strings, computer memory, x-y data, matrices, …
  - The five basic ENDF records can all be recast as LIST records
    - However, there is a lot to be said for a data type capturing the essence of its data

- **What are the basic types of data we need to store?**
  - Lessons from the formats – ENDF, ENSDF, EXFOR, RIPL, … ?
  - Lessons from GND?
  - Lessons from other types of fundamental data?
  - Lessons from programming languages, database systems, … ?

A Potential Minimal Container Set

- **ENDF is a good starting point, though with at least modernization**
- **A TEXT record is needed to provide meta-data, annotations, …**
- **The LIST record seeks to store a generic SEQUENCE of values**
  - The modern generic version of this is an N-Tuple – an ordered sequence of elements where each element can have multiple entries
  - Well suited to store, for example, resonance parameter or legendre polynomials
- **The CONT record seeks to store PARAMETERS**
  - An N-Tuple with a list of one element with one (or more) entry (or entries)
  - Well suited to store, for example, a real or complex value
- **The TAB1 record seeks to store (the ever ubiquitous) x,y data sets**
  - Some version of this is needed due to the prevalence of such data
- **The TAB2 record seeks to provide interpolation between data**
  - Given discrete distributions, some version of this is also necessary

So What Is Missing?

- **GND extends beyond 2D y(x) to include 3D y(w,x) and 4D y(v,w,x)**
  - ENDF does this with combinations of TAB2 and LIST/TAB1 functions
- **The GND table is similar to an N-Tuple or list of CONT records**
- **GND defines a matrix type**
  - ENDF does this using LIST records
- **Care must be taken not to try to come up with a data type for everything**
  - This is the purpose of higher level structures (combining appropriate primitives)
- **However, one thing that is definitely missing is a 'functional' type**
  - Procedures of how to use data should be encoded within the format
  - This allows one to ensure conformity in checking requirements and manipulations
- **And a systematic way to describe data inherent to its storage**

Human Nature Compels Us To Organize
Well, At Least If We Are Trying To Do Something Useful

What is a file system?

Once upon a time even before there were computers, people & businesses had a need to maintain certain papers and documents. So they invented Manila folders wherein they could safely organize like papers. After a while, and gathering many Manila folders on the desk, someone invented a cabinet into which these Manila folders became stored. Since each Manila folder had a specific name, it was called a file folder. Thus, the cabinet became known as a file cabinet.

-- http://answers.yahoo.com/question/index?qid=20080719211310AAkFCZU

At The Intersection Of Low- and High-Level Containers And An API To Access Them All

- **Certainly the concept that has outlived its usefulness is MAT/MF/MT/NS**

- **Every container, low- or high-, should be accessible via an appropriate hierarchical, self-describing naming scheme**
  - This should allow, and facilitate, direct access to any object
  - This should allow, and facilitate, queries into the data

- **The naming scheme should fit naturally into existing storage systems**
  - Nothing is as ubiquitous as the modern file system in computing
  - HDF and XML are the state of the art in scientific data sets and generic text
  - Any structure should be defined so as to not preclude storage in any of these three

Wrapping Up
The Path Forward…

- Settle on the set of containers
- Write up formal definitions
  - Include documentation of appropriate checking and manipulation
- Implement prototype coding to handle each container
- Work with API and high-level container groups to iterate as needed