SG43—WPEC 2017

LA-UR-17-23760

Jeremy Lloyd Conlin

May 16, 2017

Conducting SG43 Business

Jeremy Lloyd Conlin   Caleb Mattoon   Fausto Malvagi

May 16, 2017

Collaboration Space

Collaboration Space

GitHub

https://github.com/GeneralizedNuclearData

- Can have separate repositories to store documentation/code/website/etc.
    - SG43 repository contains business stuff—not code.
- Wiki for discussions and development space
- Can integrate with many other OSS tools

Making Changes

`master` branch contains the official, supported, version.

Making Changes:

1. Make branch/fork repository.
2. Make changes as desired.
3. Submit Pull Request.

Live Demo

More GitHub Help

Further Help

https://guides.github.com

- Understanding the GitHub Flow
- Hello World
- Getting your project on GitHub
- Forking Projects

Procedure for Dispute Resolution

Dispute Resolution[1]

When a disagreement occurs:

1. Try to reach consensus with *reasonable* discussion; i.e., within a reasonable amount of time in current meeting.
2. Table the decision
   2.1 Decide when decision will be made. (Suggestion: no more than 6 months)
   2.2 Make formal proposal/presentation describing each side
   2.3 Reach a consensus
3. Decision will be made by SG43 leadership:
   - Fausto Malvagi
   - Caleb Mattoon
   - Jeremy Conlin

[1] https://github.com/GeneralizedNuclearData/SG43/blob/master/DisputeResolution.md

Teleconferences and Work Assignments

Two SG43 Subcommittees

- Operate/meet independently
- Regular teleconferences to report and coordinate efforts
- Intended to last no more than 6 months (December 2017)

Reading API—Caleb

1. Develop general guidelines for API
    - Object-oriented vs. functional
    - Access routines

Physics Checks—Jeremy

1. Create list of potential physics checks
2. Prioritize list of checks
    - Importance
    - Ease of implementation
    - etc.

December–April

1. Begin implementation of API.
   Only implement enough to be able to use the API and library to perform the first 2–3 highest-priority physics checks
   Note: The physics checking does **not** belong in the library implementation.
2. Report on issues/problems/concerns
3. Modify API and library as needed
4. Implement API to be able to perform next set of physics checks
5. . . .

Using this pattern, <u>the actual use of the API will inform it's development</u>. We will know very quickly how well the API meets our needs

WPEC—May 2018

- Should have a very nice beginning of the API and one or more library implementations
- Opportunity to evaluate what our next course of action will be to accomplish goals of SG43.

NJOY Data Access

Jeremy Lloyd Conlin

May 16, 2017

NJOY2016

Legacy NJOY—NJOY2016

- Legacy NJOY is tightly coupled with ENDF format.
- Must navigate via ENDF Records and MAT,MF,MT,NS metadata at the end of every line.
- ENDF file is not stored in memory, searching/reading is very inefficient

Some ENDF routines

contio	Read/write CONT Record
tab1io	Read/write TAB1 Record
tab2io	Read/write TAB2 Record
intgio	Read/write INTG Record
lineio	Read/write one line of floating point data
tosend	Read up to (including) SEND Record
tomend	Read up to (including) MEND Record
findf(mat, mf, mt)	Find specified Section
gety1	Retrieve y1(x) from ENDF TAB1
gety2	Retrieve y2(x) from ENDF TAB2

NJOY21

NJOY21

ENDFtk

NJOY21—ENDFtk

```cpp
class ControlRecord{
  double C1();
  double C2();
  ...
};
class TextRecord{
  int MAT();
  int MF();
  int MT();
  ...
  std:: string text();
};
```

```cpp
class Material{
  File& MF(int mf)
};

class File{
  Section<MF> section(int MT);
};

template< int MF >
class Section{
  ...
};
```

NJOY21—ENDFtk

`ENDFtk` access routines are fairly straightforward.

- Read entire Tape into memory
- Methods for accessing specific pieces of data

```
Tape u235("n-092_U_235.endf");
auto fissionXS = u235.Material(9238).MF(3).section(18).XS();
```

- Tightly coupled to ENDF-6 format

Challenges:

- Tightly coupled to ENDF-6 format
- Accessing data is verbose and clunky
- Each File has a different format for its Sections
- Must access different pieces of data from different parts of the file

NJOY21

NuclearData

NJOY21—NuclearData

`NuclearData` is a library used to interact with nuclear data

- Independent of source of data
- Common in-memory data structure to be used in all parts of NJOY21 processing
- Data is organized so that related pieces of data can be accessed at the same time
- In-memory object looks an awful lot like GND hierarchy

Challenges:

- Must write an interface for every type of input data (ENDF, GND, ACE, etc.).

NJOY21

GNDtk

NJOY21-GNDtk

Utilize pugixml (http://pugixml.org) to read XML

```
pugi::xml_document doc;
doc.load_file("O-16.xml");

// Find the first reactionSuite xml_node
pugi::xml_node RSNode = doc.child("reactionSuite");
GNDtk::implementation::XMLComponentFactory XMLFact( RSNode );
nuclearData::API::Reaction captureRX = XMLFact.reaction(102);

nuclearData::API::CrossSection captureXS = captureRX.crossSection("eval");
double Qvalue = captureRX.Q();
```

GNDtk—nuclearData

- Still a work in progress—will continue to evolve.
- nuclearData is independent of GND and underlying file format

Challenges:

- Populating `nuclearData` object is too tightly coupled with GND format. An abstracted API would alleviate this problem.
- Must write code for each type of underlying data (e.g., XML, JSON, HDF5)

Final Discussion

Jeremy Lloyd Conlin Caleb Mattoon Fausto Malvagi

May 16, 2017

First Year Goals

1. Develop list of physics checks.
2. Design the “look and feel” of the API—with no expectations that the API is complete.
3. Begin implementation of library.
4. Use library—with separate checking code—to perform initial physics checks.

Subcommittees

- Two subcommittees

  Reading API led by Caleb Mattoon

  Physics Checks led by Jeremy Conlin

- First teleconferences (first week of June?)
  - API
  - Physics