ENSDF Modernization

From 80-Column Text

to JSON-Formatted Files

Benjamin Shu
National Nuclear Data Center
Brookhaven National Laboratory

ENSDF Overview

ENSDF: Evaluated Nuclear Structure Data File

Search and Retrieval

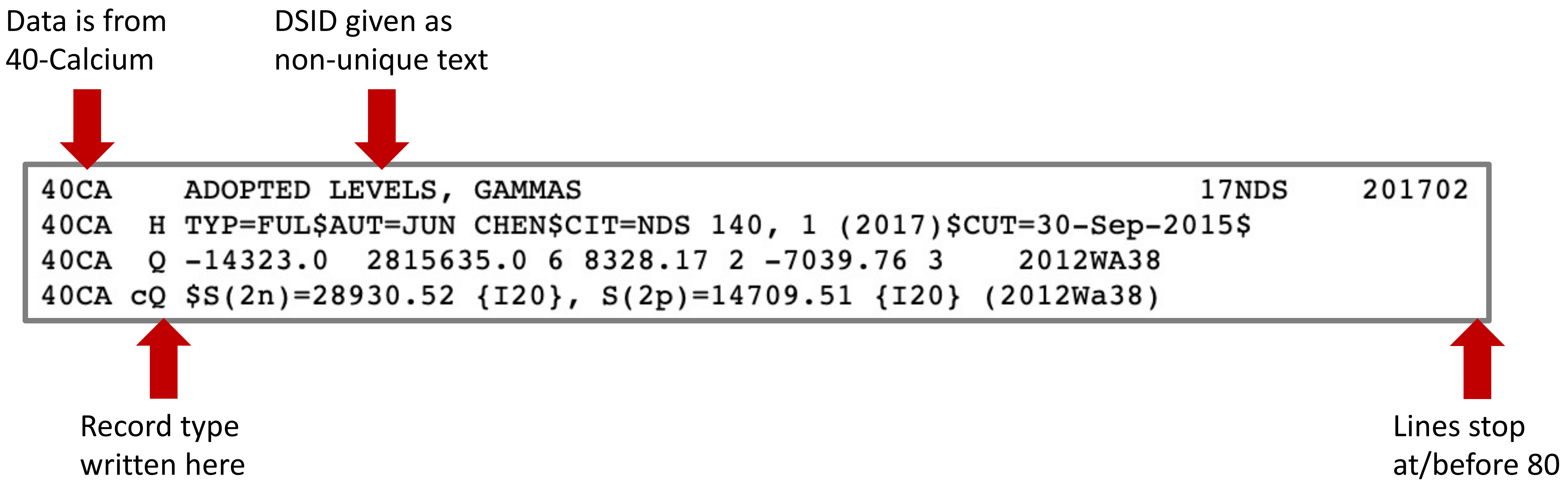
Last updated 2021-09-15

153 new datasets added/modified in the last month!

- Database of 33,000+ evaluated nuclear physics data sets
  - Structure, decay, reactions, etc.

The ENSDF Format

- Text files with maximum of 80 characters per line
- Datasets identified by mass/nuclide and a dataset ID (DSID)



```
40CA    ADOPTED LEVELS, GAMMAS                                   17NDS    201702
40CA  H TYP=FUL$AUT=JUN CHEN$CIT=NDS 140, 1 (2017)$CUT=30-Sep-2015$
40CA  Q -14323.0  2815635.0 6 8328.17 2 -7039.76 3     2012WA38
40CA cQ $S(2n)=28930.52 {I20}, S(2p)=14709.51 {I20} (2012Wa38)
```

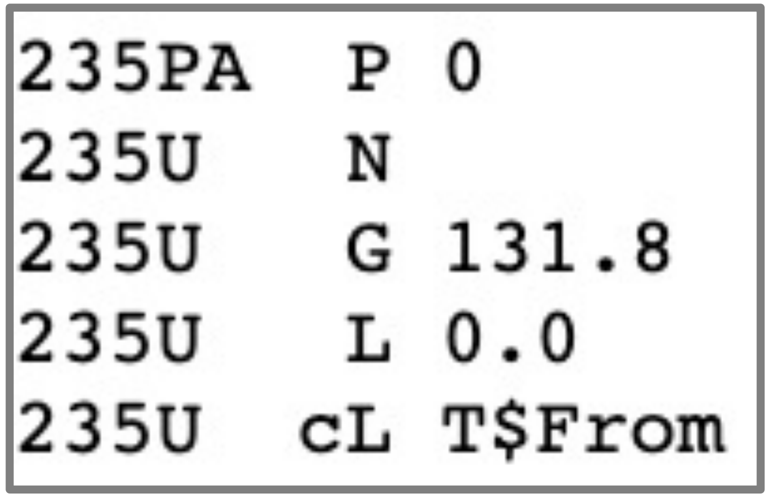
The ENSDF Format (contd.)

- Each dataset belongs to a category which defines its use:
  - **Adopted Levels, Gammas** **(4,100 datasets)**
    - Excitation state energies
    - Gamma ray emissions (from levels)
    - Q-record of common decay energies
  - **Decay** **(7,583 datasets)**
    - Parent and daughter nuclides, emitted particles
    - Normalization of radiation energies
  - **Reaction** **(20,716 datasets)**
    - Comments describing experiment (targets, beams, etc.)

The ENSDF Format (contd.)

- ENSDF datasets are composed of **records**
  - Identified using a single character in column 8
  - 10+ types, each with unique conventions
  - Written as one or more 80-column lines

P	Parent (1 line)
N	Normalization (1 line)
G	Gamma (1 line)
L	Level (2 lines)



```
235PA   P 0
235U    N
235U    G 131.8
235U    L 0.0
235U   cL T$From
```

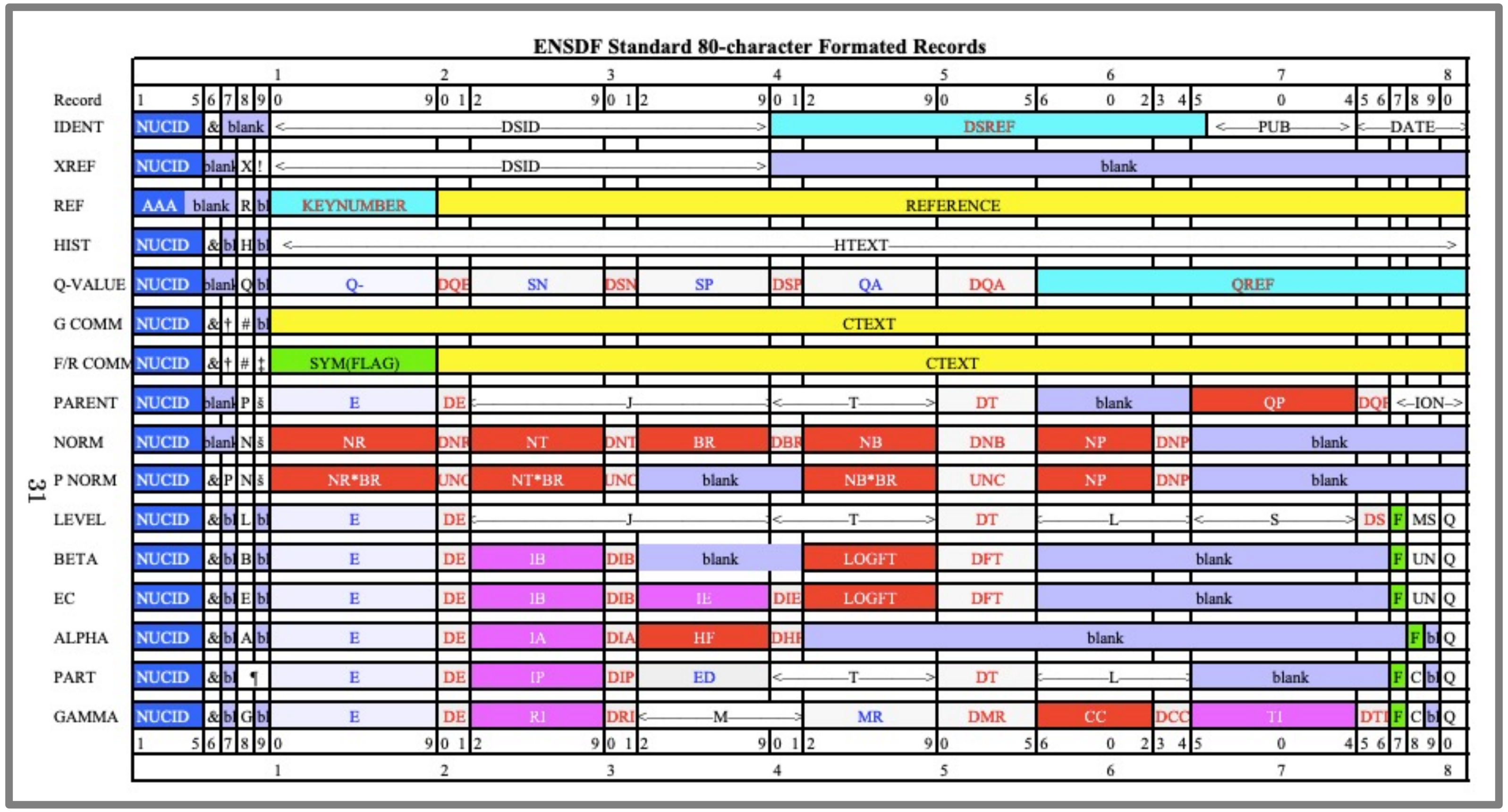
The ENSDF Format (contd.)

- Datasets in each category usually contain specific record types:

Record Type	Adopted Levels	Decay	Reaction
History	Yes	Yes	Yes
XREF	Yes	No	No
Reference	No	Yes	Yes
Comment	Yes	Yes	Yes
Level	Yes	Maybe	Maybe
Gamma	Maybe	Maybe	Maybe
Q-Values	Yes	No	No

Why JSON?

- The 80-column format saves space at the cost of user-friendliness



ENSDF Standard 80-character Formated Records

						1	2		3		4		5	6		7				8
Record	1 5	6	7	8	9	0 9	0 1	2 9	0 1	2 9	0 1	2 9	0 5	6 0 2	3 4	5 0 4	5 6	7	8 9	0
IDENT	NUCID	&	blank			<—DSID—>					DSREF					<—PUB—>	<—DATE—>			
XREF	NUCID	blank		X	!	<—DSID—>					blank									
REF	AAA	blank		R	bl	KEYNUMBER	REFERENCE													
HIST	NUCID	&	bl	H	bl	<—HTEXT—>														
Q-VALUE	NUCID	blank		Q	bl	Q-	DQE	SN	DSN	SP	DSP	QA	DQA	QREF						
G COMM	NUCID	&	†	#	bl	CTEXT														
F/R COMM	NUCID	&	†	#	‡	SYM(FLAG)	CTEXT													
PARENT	NUCID	blank		P	š	E	DE	<—J—>			<—T—>		DT	blank		QP	DQP	<—ION—>		
NORM	NUCID	blank		N	š	NR	DNR	NT	DNT	BR	DBR	NB	DNB	NP	DNP	blank				
P NORM	NUCID	&	P	N	š	NR*BR	UNC	NT*BR	UNC	blank		NB*BR	UNC	NP	DNP	blank				
LEVEL	NUCID	&	bl	L	bl	E	DE	<—J—>			<—T—>		DT	<—L—>		<—S—>	DS	F	MS	Q
BETA	NUCID	&	bl	B	bl	E	DE	IB	DIB	blank		LOGFT	DFT	blank				F	UN	Q
EC	NUCID	&	bl	E	bl	E	DE	IB	DIB	IE	DIE	LOGFT	DFT	blank				F	UN	Q
ALPHA	NUCID	&	bl	A	bl	E	DE	IA	DIA	HF	DHF	blank							F bl	Q
PART	NUCID	&	bl	¶		E	DE	IP	DIP	ED	<—T—>		DT	<—L—>		blank		F	C bl	Q
GAMMA	NUCID	&	bl	G	bl	E	DE	RI	DRI	<—M—>		MR	DMR	CC	DCC	TI	DTI	F	C bl	Q
	1 5	6	7	8	9	0 9	0 1	2 9	0 1	2 9	0 1	2 9	0 5	6 0 2	3 4	5 0 4	5 6	7	8 9	0
						1	2		3		4		5	6		7				8

31

Why JSON? (contd.)

- JSON data is easier to adapt for programming uses
  - **Web programming**
    - JSON is derived from JavaScript, which is used in >97% of all websites
  - **Object-oriented databases**
    - JSON-based documents enable flexible design

CouchDB
relax

ENSDF To JSON (Step 1)

- Most basic step is reading a single ENSDF record
- First, define a `Reader` which parses one or more lines

```
public abstract class Reader<T> {
    // Is this the right kind of record?
    public abstract boolean canRead(String line);

    // Read lines from the beginning
    public abstract int read(List<String> list_lines);

    // Read lines from a starting position
    public abstract int read(List<String> list_lines, int start);
}
```

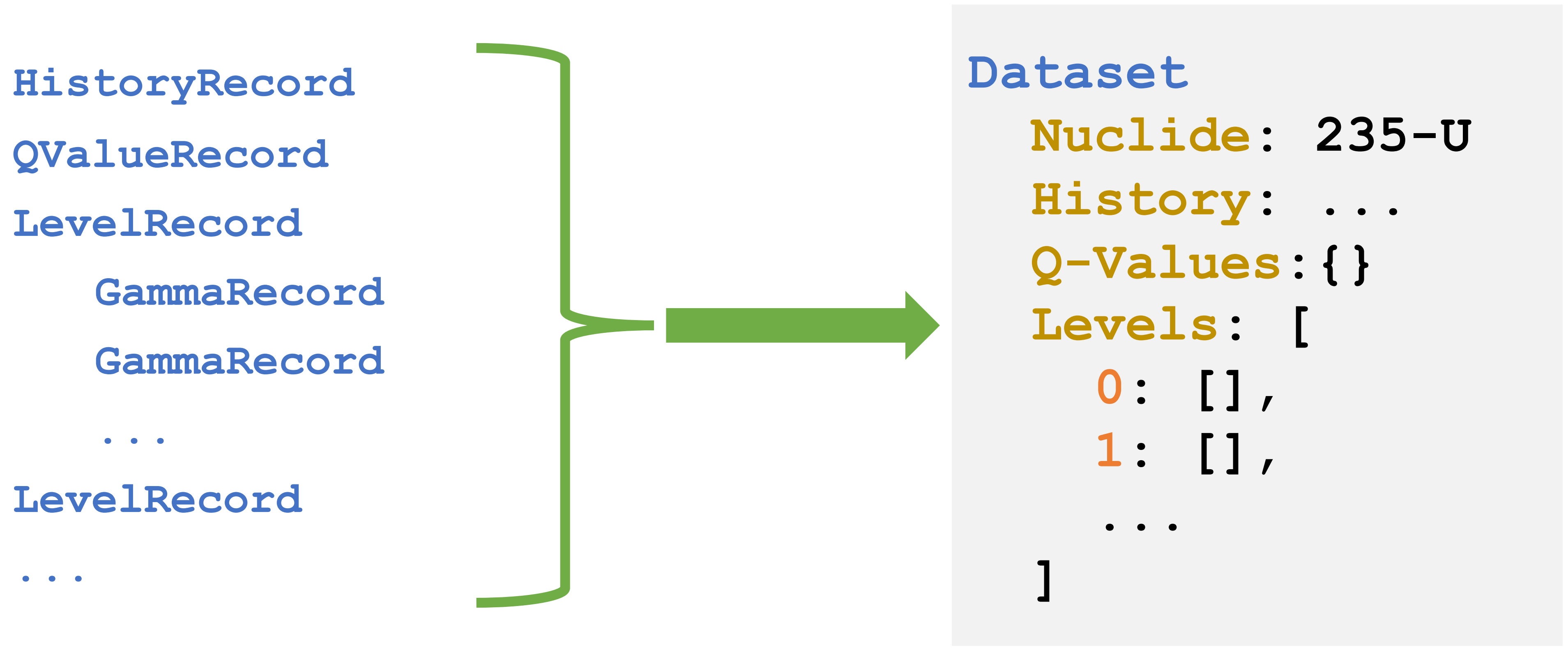
ENSDF To JSON (Step 2)

- Add specific `Reader` classes to handle each type of record
- Define `Record` objects to store parsed data

ENSDF Data		Reader		Record
`235U  H ...`	→	`HistoryReader`	→	`HistoryRecord`
`235U  C ...`	→	`CommentReader`	→	`CommentRecord`
`235U  X ...`	→	`XREFReader`	→	`XREFRecord`
`235U  L ...`	→	`LevelReader`	→	`LevelRecord`
`235U  G ...`	→	`GammaReader`	→	`GammaRecord`
...		...		...

ENSDF To JSON (Step 3)

- Assemble a completed `Dataset` using `Record` objects



```
HistoryRecord
QValueRecord
LevelRecord
    GammaRecord
    GammaRecord
    ...
LevelRecord
...
```

```
Dataset
  Nuclide: 235-U
  History: ...
  Q-Values:{}
  Levels: [
    0: [],
    1: [],
    ...
  ]
```

ENSDF To JSON (Step 4)

- Print the completed `Dataset` as a JSON-formatted text file

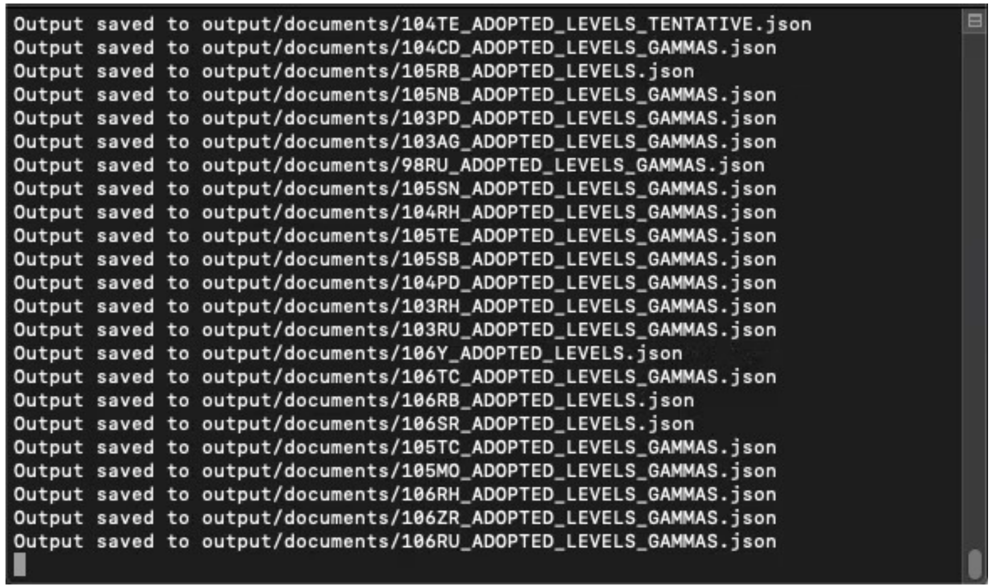
```
Dataset
  Nuclide: 235-U
  History: ...
  Q-Values:{}
  Levels: [
    0: [],
    1: [],
    ...
  ]
```

```
{
  “nuclide”: ”235U”,
  “history”: { ... },
  “qValues”: { ... },
  “levels”: {
    “0”: {
      “gammas”: {}
    },
    ...
  }
}
```

Theory to Practice

- Java library used to build an `ENSDFToJSON` executable, which:
  - Retrieves text for all ENSDF datasets
  - Builds `Dataset` objects from each file
  - Prints those `Dataset` objects as JSON files

- Can convert 33,385 ENSDF files in around 7-8 minutes
  - The power of multithreading!



```
Output saved to output/documents/104TE_ADOPTED_LEVELS_TENTATIVE.json
Output saved to output/documents/104CD_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/105RB_ADOPTED_LEVELS.json
Output saved to output/documents/105NB_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/103PD_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/103AG_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/98RU_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/105SN_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/104RH_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/105TE_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/105SB_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/104PD_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/103RH_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/103RU_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/106Y_ADOPTED_LEVELS.json
Output saved to output/documents/106TC_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/106RB_ADOPTED_LEVELS.json
Output saved to output/documents/106SR_ADOPTED_LEVELS.json
Output saved to output/documents/105TC_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/105MO_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/106RH_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/106ZR_ADOPTED_LEVELS_GAMMAS.json
Output saved to output/documents/106RU_ADOPTED_LEVELS_GAMMAS.json
```

Ongoing Development

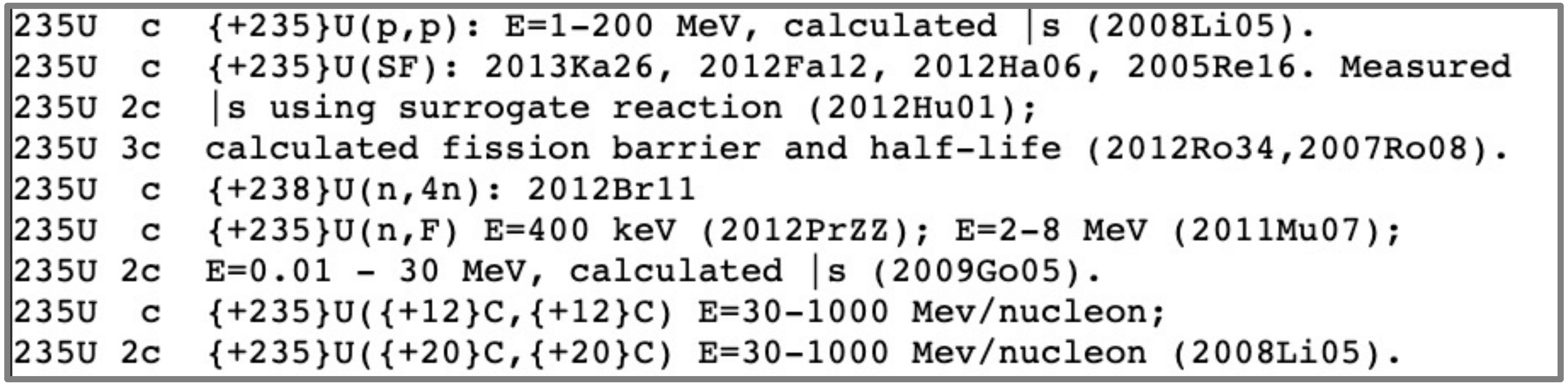
- We have JSON files – now what?
  - Building and testing schema validation
    - JSON files need to be checked for "correct" formatting
    - Will be needed for future evaluations

  - Currently finalizing format for the Adopted Levels
    - Adding details, field names, etc.
    - Being handled by a subprogram named **`ConvertAdopted`**

  - Re-thinking ENSDF as an object-oriented database
    - Prototype designs using CouchDB

Lingering Questions

- Comments
  - Converted to LaTeX through Java-NDS
  - Often contain valuable information which is difficult to extract
  - Will require natural language processing (i.e. machine learning)



```
235U  c  {+235}U(p,p): E=1-200 MeV, calculated |s (2008Li05).
235U  c  {+235}U(SF): 2013Ka26, 2012Fa12, 2012Ha06, 2005Re16. Measured
235U 2c  |s using surrogate reaction (2012Hu01);
235U 3c  calculated fission barrier and half-life (2012Ro34,2007Ro08).
235U  c  {+238}U(n,4n): 2012Br11
235U  c  {+235}U(n,F) E=400 keV (2012PrZZ); E=2-8 MeV (2011Mu07);
235U 2c  E=0.01 - 30 MeV, calculated |s (2009Go05).
235U  c  {+235}U({+12}C,{+12}C) E=30-1000 Mev/nucleon;
235U 2c  {+235}U({+20}C,{+20}C) E=30-1000 Mev/nucleon (2008Li05).
```

What Comes Next?

- ENSDF modernization is a 3-year project
  - Currently at the end of Year 1
  - Objective: Make JSON draft documents public in Year 2

- Database to be re-designed after finalizing formats

- JSON files will be distributed through the ENSDF website
  - Will be available along with original 80-column format
  - Possible RESTful API?