https://doi.org/10.61092/iaea.hz1z-0dx3

IAEA-NDS-0244
Rev. 2025/01

INTERNATIONAL ATOMIC ENERGY AGENCY

NUCLEAR DATA SERVICES

DOCUMENTATION SERIES OF THE IAEA NUCLEAR DATA SECTION

ForEX: Utility Codes for EXFOR

Naohiko Otuka
IAEA Nuclear Data Section, Vienna, Austria

January 2025

IAEA Nuclear Data Section, Vienna International Centre, A-1400 Vienna, Austria

Note:

The IAEA-NDS-reports should not be considered as formal publications. When a nuclear data library is sent out by the IAEA Nuclear Data Section, it will be accompanied by an IAEA-NDS-report which should give the data user all necessary documentation on contents, format and origin of the data library.

IAEA-NDS-reports are updated whenever there is additional information of relevance to the users of the data library.

For citations care should be taken that credit is given to the author of the data library and/or to the data centre which issued the data library. The editor of the IAEA-NDS-report is usually not the author of the data library.

Neither the originator of the data libraries nor the IAEA assume any liability for their correctness or for any damages resulting from their use.

96/11

Citation guideline:

When quoting the EXFOR Utility Codes in a publication this should be done in the following way:

N. Otuka, "ForEX: Utility Codes for EXFOR", report IAEA-NDS-0244 Rev. 2025/01, International Atomic Energy Agency, 2024 (https://doi.org/10.61092/iaea.hz1z-0dx3).

IAEA-NDS-0244
Rev. 2025/01

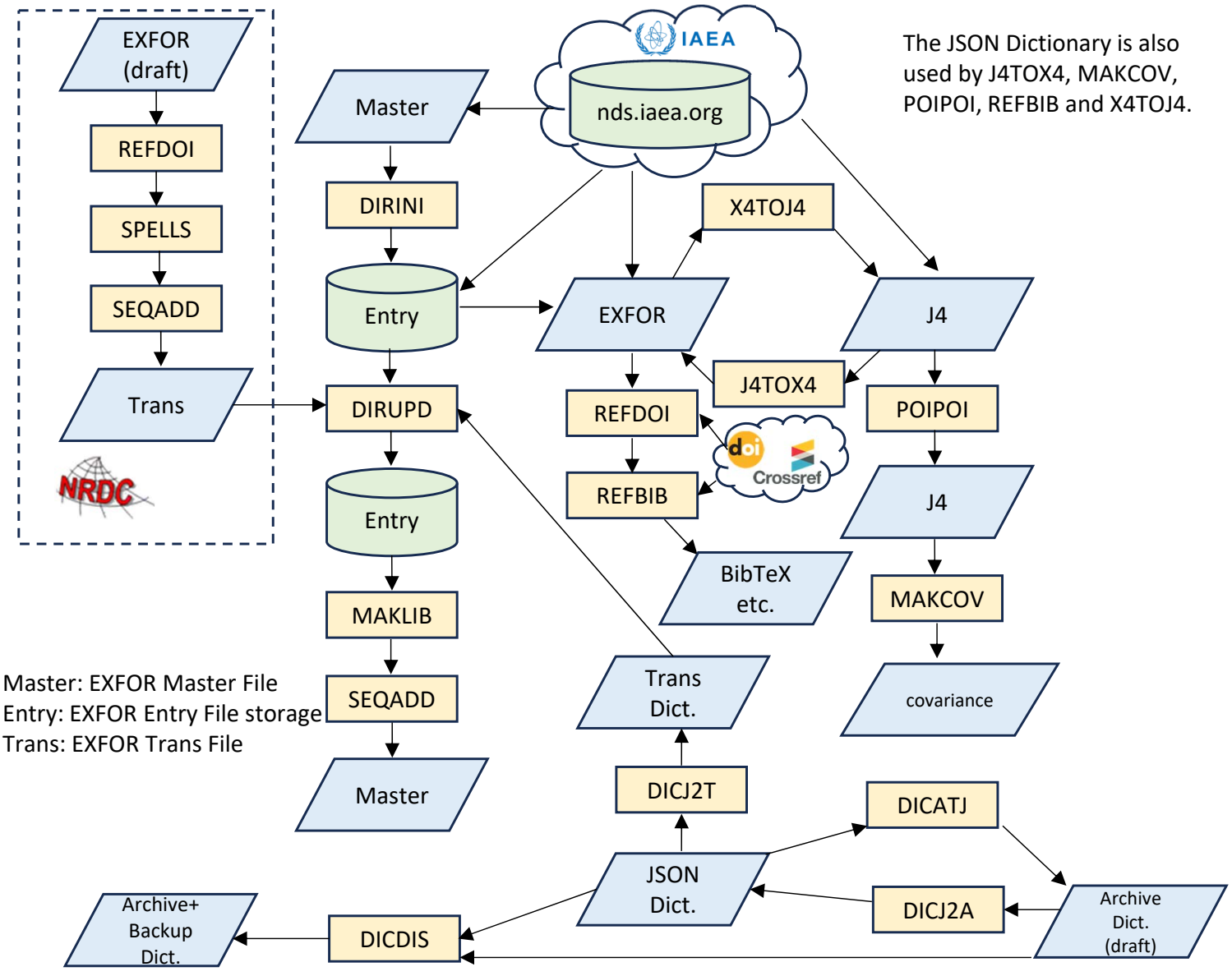
ForEX: Utility Codes for EXFOR

Naohiko Otuka
IAEA Nuclear Data Section, Vienna, Austria

Abstract

Descriptions are given for a package of utility codes operating on the experimental nuclear reaction data files in the EXFOR format. This program package is written in Python and may be downloaded from the NRDC software website (http://nds.iaea.org/nrdc/nrdc_sft/).

January 2025


EXFOR (draft)
REFDOI
SPELLS
SEQADD
Trans
NRDC
IAEA
nds.iaea.org
Master
DIRINI
Entry
DIRUPD
Entry
MAKLIB
SEQADD
Master
X4TOJ4
EXFOR
J4
J4TOX4
REFDOI
POIPOI
REFBIB
J4
BibTeX etc.
MAKCOV
covariance
Trans Dict.
DICJ2T
DICATJ
JSON Dict.
DICJ2A
Archive Dict. (draft)
Archive+ Backup Dict.
DICDIS
The JSON Dictionary is also used by J4TOX4, MAKCOV, POIPOI, REFBIB and X4TOJ4.
Master: EXFOR Master File
Entry: EXFOR Entry File storage
Trans: EXFOR Trans File

Introduction

The Utility Codes for EXFOR Utility (ForEX) are written to process EXFOR Entry files and EXFOR/CINDA Dictionary files. Currently, the following 17 codes (Python scripts) are included in this package:

- DIC227 Produce Archive Dictionary 227 from a NUBASE file.
- DICA2J Convert Archive dictionaries to a JSON Dictionary.
- DICDIS Prepare Archive and Backup dictionaries for distribution.
- DICJ2A Convert a JSON Dictionary to Archive dictionaries.
- DICJ2T Convert a JSON Dictionary to a Transmission dictionary.
- DIRINI Split an EXFOR library tape into EXFOR entry files.
- DIRUPD Update the EXFOR entry files with an EXFOR transmission tape.
- EXTMUL Extraction of a dataset from a multiple reaction formalism subentry.
- J4TOX4 Convert a J4 file to an EXFOR file.
- MAKCOV Produce a data table and covariance matrix from a J4 file.
- MAKLIB Merge EXFOR entry files into a single library tape.
- POIPOI Remove pointers from a J4 file.
- REFBIB Extract bibliography of reference by using DOI.
- REFDOI Obtain DOI for articles registered in CrossRef.
- SEQADD Add record identification to an EXFOR file.
- SPELLS Check English spell in free text in EXFOR format.
- X4TOJ4 Convert an EXROF file to a J4 file.

("J4" means EXFOR in JSON.)

This package is distributed in a zipped form from the NRDC software website (https://nds.iaea.org/nrdc/nrdc_sft/).

This document explains how to use these codes. Users need to install Python3 in their environments prior to run these codes. Any comments on the use of the codes, including difficulties encountered or any possible bug reports and suggestions are welcome.

Optional arguments available in all codes

- `-h` Display help information
- `-v` Display the version
- `-f` Never prompt

Acknowledgements

The developer would like to thank David Brown, Oscar Cabellos, Georg Schnabel and Nicolas Soppera for their comments and proposals.

History (major revisions only)

2023-10-23:

- First release of 4 scripts (DIRINI, DIRUPD, MAKLIB, SEQADD)

2023-11-02:

- First release of IAEA-NDS-0244.

2024-05-03:

- First release of 6 scripts (DIC227, DICA2J, DICDIS, DICJ2A, DICJ2T, SPELLS)

2024-06-25:

- Addition of -c option to MAKLIB.
- Update of DICA2J to implement format changes of Dictionaries 25, 209 and 227 concluded in the NRDC 2024 meeting (C12 and C13).

2024-10-07:

- First release of 4 scripts (J4TOX4, MAKTAB, POIPOI, X4TOJ4).

2024-11-21:

- First release of EXTMUL. MAKTAB renamed into MAKCOV.

2025-01-05:

- First release of REFBIB and REFDOI.

DIC227

This code reads a NUBASE file and a supplemental dictionary file (compilation of properties of elementary particles and natural elements in the Archive Dictionary format), and convert them to Archive Dictionary227.

Input files

- NUBASE file[1] (`-i`)
- supplemental file (`-s`)

Output file

- Archive Dictionary 227 file (`-o`)

Arguments

- `-i` *`file_nubase`* input NUBASE file
- `-s` *`file_suppl`* input supplemental dictionary file
- `-o` *`file_arc227`* output archive dictionary file

Example

Convert a NUBASE file *nubase_4.mas20.txt* and a supplemental dictionary file *dict_arc_sup.227* to the Archive Dictionary 227 *dict_arc_new.227*:

```
python3 x4_dic227.py -i nubase_4.mas20.txt -s dict_arc_sup.227
-o dict_arc_new.227
```

[1] The NUBASE file is distributed from the Atomic Mass Data Center (https://amdc.impcas.ac.cn/, https://www.anl.gov/phy/atomic-mass-data-resources, https://nds.iaea.org/amdc/).

DICA2J

This code reads Archive Dictionaries and convert them to a JSON Dictionary.

Input file

- Archive Dictionaries (`-i`)

Output file

- JSON Dictionary (`-o`)

Arguments

- `-n` *`dict_ver`* dictionary version (transmission ID)
- `-i` *`dir_archive`* directory of input Archive Dictionaries
- `-o` *`dir_json`* directory of output JSON Dictionary

Example

Convert Archive Dictionaries *input/dict_arc.top, input/dict_arc_new.001* etc. to a JSON Dictionary *json/dict_9131.json*.

```
python3 x4_dica2j.py -n 9131 -i input -o json
```

DICDIS

This code reads Archive and JSON Dictionaries and process them for distribution after removal of records with the alteration flag D (deletion) and updating the year + month field (YYYYMM). At the same time, this code also produces the Backup Dictionary.

Input files

- Archive Dictionaries (`-a`)
- JSON Dictionary (`-j`)

Output files

- Archive, Backup and JSON Dictionaries for distribution (`-o`)

Arguments

- `-n` *`dict_ver`* dictionary version (transmission ID)
- `-a` *`dir_archive`* directory of input Archive Dictionaries
- `-j` *`dir_json`* directory of input JSON Dictionary
- `-o` *`dir_dist`* directory of output dictionaries

Example

Read Archive and JSON Dictionaries *input/dict_arc.top, input/dict_arc_new.001* etc. and *json/dict_9131.json*, process them for distribution, and output them under the same names but under the directory *dist*. At the same time, it also produces the Backup Dictionary *dist/dan_back_new.9131*.

```
python3 x4_dicdis.py -n 9131 -a input -j json -o dist
```

DICJ2A

This code reads JSON Dictionary and convert it to Archive Dictionaries.

Input file

- JSON Dictionary (`-i`)

Output files

- Archive Dictionaries (`-o`)

Arguments

- `-n` *`dict_ver`* dictionary version (transmission ID)
- `-i` *`dir_json`* directory of input JSON Dictionary
- `-o` *`dir_archive`* directory of output Archive Dictionaries

Example

Convert a JSON Dictionary *json/dict_9131.json* to Archive Dictionaries *output/dict_arc.top, output/dict_arc_new.001* etc..

```
python3 x4_dicj2a.py -n 9131 -i json -o output
```

DICJ2T

This code reads JSON Dictionary and convert it to a Transmission (TRANS) Dictionary.

Input file

- JSON Dictionary (`-i`)

Output file

- TRANS Dictionary (`-o`)

Arguments

- `-n` *`trans_ID`* dictionary version (transmission ID)
- `-i` *`dir_json`* directory of input JSON Dictionary
- `-o` *`dir_trans`* directory of output TRANS Dictionary

Example

Convert JSON Dictionary *json/dict_9131.json* to TRANS Dictionary *dist/trans.9131*.

```
python3 x4_dicj2t.py -n 9131 -i json -o dist
```

DIRINI

This code reads an EXFOR library tape (e.g., a master file[2]) in EXFOR formats with a `MASTER`, `LIB` or `REQUEST` record as the first record, splits it into entries, and saves each entry file in an entry storage. It initialises the storage (i.e, namely delete the files in the storage directory) at the beginning of processing.

Input file

- library tape (`-l`)

Output file

- entry files (`-d`)
- log file (`-g`)

Arguments

- `-c` Delete (1) trailing blanks in col. 12-66, (2) line sequential number (col.67-80) and (3) N2 of ENDBIB/ENDCOMMON/ENDSUBENT/ENDENTRY.
- `-l` *`file_lib`* input library tape
- `-d` *`dir_storage`* directory of output entry storage
- `-g file_log` output log file

Example

Initialise the entry storage directory *entry* by loading the input library tape *lib/library.txt* (without elimination of the record identification and bookkeeping if the input library has them) and record it in *x4_dirupd.log*:

```
python3 x4_dirini.py -l lib/library.txt -d entry -g x4_dirupd.log
```

At the end of processing, one obtains entry files under *entry*/1/, *entry*/2/, etc. and a log file with a new line like

```
Seq. Update date/time           Trans(N1) Trans(N2)  Centre Tape
   0 2023-10-04 00:48:30.849567 0001      20231004          lib/library.txt
```

Note

A storage initialized by DIRINI with the most recent EXFOR Master File and updated by DIRUPD with all TRANS files released after the EXFOR Master File is equivalent to the zipped version of the EXFOR Entry Files distributed from the NRDC website (https://nds.iaea.org/nrdc/exfor-master/entry/).

[2] The NRDC release the EXFOR Master File on an annual basis on the NRDC website (https://nds.iaea.org/nrdc/exfor-master/).

DIRUPD

This code reads a trans tape[3] (starting from the `TRANS` record) and adds or updates the entry files in the local storage.

Input files

- trans tape (`-t`)
- entry files (`-d`)
- log file (`-g`)

Output files

- entry files (`-d`)
- log file (`-g`)

Arguments

- `-c` (Same as DIRINI. Use this option if you use it for DIRINI.)
- `-t` *`file_trans`* input trans tape
- `-d` *`dir_storage`* (Same as DIRINI)
- `-g file_log` input/output log file

Example

Update of the entry storage *entry* by a tape *trans/trans.txt*, and record it in *x4_dirupd.log*:

```
python3 x4_dirupd.py -t trans/trans.txt -d entry -g x4_dirupd.log
```

At the end of processing, one obtains new and/or updated entry files under *entry*/1/ etc. and a log file with a new line like

```
Seq. Update date/time           Trans(N1) Trans(N2)  Centre Tape
   0 2023-10-04 00:48:30.849567 0001      20231004           lib/library.txt
   1 2023-10-04 00:48:31.725707 1234      20160121   NNDC    trans.txt
```

[3] The NRDC assembles a set of new and revised EXFOR entries in a TRANS file on a regular basis, and it is released on the NRDC website (https://nds.iaea.org/nrdc/exfor-master/trans/)

EXTMUL

This code reads an EXFOR file containing subentries with the multiple reaction formalism, and write an extracted dataset as an EXFOR file. Three other Python scripts files for J4TOX4, POIPOI and X4TOJ4 must be placed together with EXTMUL in the same directory.

Input files

- EXFOR file (`-i`)
- JSON Dictionary (`-d`)

Output file

- EXFOR file (`-o`)

Arguments

- `-i` *`file_x4`* input EXFOR file
- `-d` *`file_dict`* input JSON Dictionary
- `-e` *`data_id`* EXFOR dataset ID for extraction (or "all" to extract all datasets)
- `-o` *`file_x4`* output EXFOR file

Example

Extract a dataset 23756.002.3 from an EXFOR file *exfor.txt* to an EXFOR file *exfor_out.txt* with the dictionary *dict_9131.json*:

```
python3 x4_extmul.py -i exfor.txt -d dict_9131.json -e 23756.002.3 -o
exfor_out.txt
```

Extract all datasets.1 from an EXFOR file *exfor.txt* to an EXFOR file *exfor_out.txt* with the dictionary *dict_9131.json*:

```
python3 x4_extmul.py -i exfor.txt -d dict_9131.json -e all -o
exfor_out.txt
```

Note that the first subentry (e.g., 23756.001) must be always present in the input EXFOR file.

J4TOX4

This code reads a J4 file and convert it to an EXFOR file (starting from an `ENTRY`, `MASTER`, `REQUEST` or `TRANS` record).

Input files

- J4 file (`-i`)
- JSON Dictionary (`-d`)

Output file

- EXFOR file (`-o`)

Arguments

- `-i` *`file_j4`* input J4 file to read
- `-d` *`file_dict`* directory of the input JSON Dictionary
- `-o` *`file_x4`* output EXFOR file

Example

Convert a J4 file *exfor.json* to an EXFOR file *exfor.txt* with the dictionary *dict_9131.json*:

```
python3 x4_j4tox4.py -i exfor.json -d dict_9131.json -o exfor.txt
```

MAKCOV

This reads a J4 file processed by `POIPOI`, and print a four-column (x, Δx, y, Δy) table and correlation coefficients (upper triangle matrix).

Input files

- J4 file processed by `POIPOI` (`-i`)
- "HED file" (`-j`). See Appendix of this manual for its details.
- JSON Dictionary (`-d`)

Output files

- covariance file (`-o`)
- log file (`-g`)

Arguments

- `-s` treat the dataset as a shape (normalization free) dataset
- `-r` print Δy/y (%) rather than Δy
- `-i` *`file_j4`* input J4 file
- `-j` *`file_hed`* input "HED file"
- `-d` *`file_dict`* input JSON Dictionary
- `-e` *`data_id`* EXFOR Dataset ID[4]
- `-o` *`file_cov`* output covariance file
- `-g` *`file_log`* output log file
- `-l` *`x_min`* lower boundary of independent variable to process
- `-u` *`x_max`* upper boundary of independent variable to process

Example

Read a J4 file *exfor.json*, HED file *exfor_hed.txt*, JSON dictionary *dict_9131.json*, and create a table file *x4_makcov_out.txt* for a dataset *22742.004.1*:

```
python3 x4_makcov.py -i exfor.json -j exfor_hed.txt -d dict_9131.json
-e 22742.004.1 -o x4_makcov_out.txt
```

[4] To identify the corresponding dataset ID (e.g., 22742.004.1) in the J4 file, the J4 file must be created by `POIPOI` without the option -d (delete pointer).

MAKLIB

This reads and combines the entry files in the entry storage and create a single library tape.

Input file

- entry files (`-d`)

Output file

- library tape (`-l`).

Arguments

- `-a` Add "19" to two-digit year in N2 of ENTRY/SUBENT/NOSUBENT.
- `-c` (Same as DIRINI.)
- `-n` Exclude dictionaries
- `-d` *`dir_storage`* (Same as DIRINI)
- `-l` *`file_lib`* output library tape
- `-i` *`tape_ID`* Tape ID (integer printed at cols 12-22 of the first record)

Example

Create a library tape *lib/library.txt* by merging entry files in the storage *entry* with the tape ID *0001*:

```
python3 x4_maklib.py -d entry -l library.txt -i 0001
```

This operation does not add a new line in the log file.

Note: The format of the output library tape depends on the format of the files in the entry storage. If user maintains the entry files in the storage without record identifications etc. (e.g., with -c option of DIRINI and DIRUPD), then the produced library tape also does not have them. The record identifications can be added later by processing the output library tape by SEQADD.

POIPOI

This reads a J4 file, extract the information relevant to a particular dataset, and create a J4 file. It removes the pointer structure in the input J4 file. If wish, one can (1) merge the information originally compiled in the common (001) subentry and the data subentry, and (2) select the keywords (e.g., TITLE, AUTHOR) to be kept in the output J4 file.

Input files

- J4 file (`-i`)
- JSON Dictionary (`-d`)

Output file

- J4 file (`-o`)

Arguments

- `-k` *`keywords`* BIB keywords to keep[5]
- `-p` Delete the pointer
- `-1` Keep the common (001) and data subentries separately
- `-i` *`file_inp`* input J4 file
- `-e` *`data_ID`* EXFOR Dataset ID (or "all" to process all datasets in the input)
- `-o` *`file_out`* output J4 file

Example

Read a J4 file *exfor.json*, and JSON dictionary *dict_9131.json*, and create another J4 file *exfor_poi.json* for a dataset *22742.004.1*: by keeping the BIB records under AUTHOR and REFERENCE:

```
python3 x4_poipoi.py -i exfor.json -d dict_9131.json -e 22742.004.1
-o exfor_poi.json
```

Same as above, but remove all data descriptions other than those under AUTHOR and REFERENCE:

```
python3 x4_poipoi.py -i exfor.json -d dict_9131.json -e 22742.004.1
-o exfor_poi.json -k AUTHOR REFERENCE
```

[5] The REACTION information is always kept.

REFBIB (New!)

This code converts an EXFOR reference code to DOI and vice versa. Then it extracts various bibliography related items (e.g., title, authors) from the metadata deposited in DOI registration agencies, and outputs them in various format. Currently this code supports only DOIs of journal articles registered in CrossRef.

Execution requires Python modules **pylatexenc**, **pyspellchecker** and **requests**, which may be installed by the following command if pip (a standard Python package manager) is installed in your computer:

```
pip install pylatexenc

pip install pyspellchecker

pip install requests
```

Input file

- JSON Dictionary (`-d`)

Output file

- Bibliography file (`-o`)

Arguments

- `-s` Strip accent symbols in output
- `-a fauthor` Family name of the first author (optional)
- `-i ref_inp` EXFOR reference code or DOI
- `-d file_dict` input JSON Dictionary
- `-o file_out` output bibliography file
- `-r format` output format (doi, piped, exfor, bibtex, json or xml)
- `-e email` your email address (for notification to CrossRef)

Example

For the EXFOR reference code *J,NDS,120,272,2014*, read a JSON dictionary *dict_9131.json* and email address *email@address.com* and write a BibTeX file *exfor.bib*:

```
python3 x4_refbib.py -i J,NDS,120,272,2014 -d dict_9131.json -o
exfor.bib -r bibtex -e email@address.com
```

REFDOI (New!)

This code reads an EXFOR file, and check presence of a DOI for each article coded under REFERENCE, REL-REF and MONIT-REF when the article is from a journal volume registered in CrossRef. Three Python scripts files for REFCHK, X4TOJ4 and REFBIB must be placed in the same directory.

Execution of REFDOI requires Python modules **pylatexenc**, **pyspellchecker** and **requests**. See the page for REFDOI for their installation.

Input files

- EXFOR file (`-i`)
- JSON Dictionary (`-d`)
- Email address (`-e`)

Output file

- DOI file (`-o`)

Arguments

- `-i` *`file_x4`* input EXFOR file
- `-d` *`file_dict`* input JSON Dictionary
- `-o` *`file_out`* output DOI file
- `-e email` your email address (for notification to CrossRef)

Example

Check reference codes included in *exfor.txt* with the dictionary *dict_9131.json*:

```
python3 x4_refdoi.py -i exfor.txt -d dict_9131.json -o
x4_refdoi_out.txt -e email@address.com
```

SEQADD

This code adds and/or updates record identifications (cols.67-79) and bookkeeping information such as N1 and N2 of BIB and ENDBIB. records. (Similar to ORDER developed at NNDC and ZORDER developed at NDS).

Input file

- Entry, trans or library tape (-i)

Output file

- Entry, trans or library tape (-o)

Arguments

- -m Do not add "19" to two-digit year in N2 of ENTRY/SUBENT/NOSUBENT, and do not alter N2 of ENDBIB/ENDCOMMON/ENDDATA/ENDSUBENT/ENDENTRY/ENDSUBDICT records.
- -i *file_inp* input EXFOR file
- -o *file_out* output EXFOR file

Example

Process an EXFOR file *exfor.txt* by adding and updating the record identification and bookkeeping information, and print the output in *exfor_ord.txt*.

```
python3 x4_seqadd.py -i exfor.txt -o exfor_ord.txt
```

SPELLS

This code checks English spells in the free text field in the EXFOR format. It checks each set of lower characters in free text (i.e., the first word of a sentence is not checked). The default dictionary does not know nuclear physics technical terms, and the user should add the to the dictionary to minimize the output.

Execution requires a Python module **pyspellchecker**, which may be installed by the following command if pip (a standard Python package manager) is installed in your computer:

```
pip install pyspellchecker
```

Input files

- Entry, trans or library tape (`-i`)
- Dictionary listing known technical words (`-d`)

  Example of the dictionary file (a plain text file to be updated by the user by adding more technical terms etc.)

```
atm
deadtime
decoupler
epithermal
fluence
linac
nonuniformity
subentry
```

Output file

- list of typos (`-l`)

Arguments

- `-i` *`file_x4`* input EXFOR file
- `-d` *`file_dict`* input known word dictionary
- `-o` *`file_output`* output typo list file

Example

Check spells in an EXFOR entry file *exfor.txt* with a dictionary *x4_spells.dic* and record the checking result in *x4_spells.log*.

```
python3 x4_spells.py -i exfor.txt -d x4_spells.dic -l x4_spells.log
```

X4TOJ4

This code reads an EXFOR file (starting from an `ENTRY`, `MASTER`, `REQUEST` or `TRANS` record) and convert it to a J4 file. One can select the keywords (e.g., TITLE, AUTHOR) to be kept in the J4 file.

Input files

- EXFOR file (`-i`)
- JSON Dictionary (`-d`)

Output file

- J4 file (`-o`)

Arguments

- `-c` Check record identification of the system identifiers
- `-k` *`keywords`* BIB keywords to be kept[6]
- `-g` Keep the flag at column 80 of each record of the DATA sections
- `-s` Keep real numbers as strings
- `-i` *`file_x4`* input EXFOR file
- `-d` *`file_dict`* input JSON Dictionary
- `-o file_j4` output J4 file

Example

Read an EXFOR file *exfor.txt* and JSON dictionary *dict_9131.json* and write a J4 file *exfor.json*:

```
python3 x4_x4toj4.py -i exfor.txt -d dict_9131.json -o exfor.json
```

Same as above, but remove all data descriptions other than those under AUTHOR, REFERENCE and REACTION:

```
python3 x4_x4toj4.py -i exfor.txt -d dict_9131.json -o exfor.json -k
AUTHOR REFERENCE REACTION
```

[6] The REACTION information is always kept.

Appendix 1: HED file used in MAKTAB

(See also N. Otuka, O. Iwamoto, INDC(SEC)-0112(Rev.) Sect.2.3)

Basic structure

The HED file defines the independent (x) and dependent (y) variables for tabulation by MAKTAB. It may have the following lines:

- L.1: Heading for the lower boundary of the independent variable.
- L.2: Heading for the upper boundary of the independent variable (may be repetition of the heading on L.1).
- L.3: Heading for the uncertainty or resolution of the independent variable (optional). The number under this heading is ignored if it is smaller than the difference of the numbers under the headings specified in L.1 and L.2.
- L.4: Heading for the dependent variable. The heading specified in L.4 is typically `DATA` or `DATA-CM`.
- L.5: Heading for the reference variable (optional). The heading specified in L.5 is typically `MONIT`. The number under this heading is used when conversion of the number to or from the fractional (%) uncertainty is required.
- L.6: Heading for the total uncertainty in the dependent variable (optional if L.7 is given). The heading specified in L.6 is typically `ERR-T` or `DATA-ERR`.
- L.7: Heading for the 1st partial uncertainty in the dependent variable (optional if L.6 is given). The heading specified in L.7 and below is typically `ERR-S`, `ERR-1`, `ERR-2` etc.
- L.8: Heading for the 2nd partial uncertainty in the dependent variable (optional)

etc.

The column 1 is reserved for a flag. It must be empty in Line 1 to 6 must be empty. In Line 7 and below, this column can be empty or

- "S": indicates the uncertainty must be ignored when the dataset is treated as a shape dataset (MAKTAB with option `-s`)
- "#": indicates it is a comment line (always ignored).

The column 2 is for the first character of a heading if a heading is given on the line.

When only a range of the partial uncertainty is given under ERR-ANALYS, its lower boundary is treated as a constant of the dataset. The user may add an extra uncertainty following an ad-hoc heading such as `ERR-A`, `ERR-B`.

Tabulation without estimation of correlation coefficients

When the only tabulation in the four-column (x, Δx, y, Δy) structure is necessary, L.5 always has a correlation flag "+". If the partial uncertainties are in EXFOR without the total uncertainty, the partial uncertainties under the heading in L.6 and below must be flagged by "-". Δy is calculated by adding these partial uncertainties in % quadratically.

Example 1

Δx is calculated by the difference between EN-MIN and EN-MAX. Δy is taken from ERR-T.

```
1: EN-MIN
2: EN-MAX
3:
4: DATA
5:
6: ERR-T       +
```

Example 2

Δx is taken from EN-ERR. Δy is obtained by the quadrature sum of ERR-S, ERR-1 and ERR-2.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6:             +
7: ERR-S       0
8: ERR-1       0
9: ERR-2       0
```

Example 3

Δx is always zero. Δy is obtained by the quadrature sum of ERR-S, ERR-1 and MONIT-ERR. The MONIT-ERR coded in other than % (e.g., in barn) is divided by MONIT for conversion to the fractional (%) uncertainty. The flag "S" on the L.9 indicates that the MONIT-ERR value is ignored when the dataset is treated as a shape dataset (i.e., MAKTAB is used with the option -s).

```
1: EN
2: EN
3:
4: DATA
5: MONIT
6:             +
7: ERR-S       0
8: ERR-1       0
9:SMONIT-ERR   0
```

Tabulation with estimation of correlation coefficients

The correlation coefficients are calculated on the assumption that each partial uncertainty is fully correlated or uncorrelated between two data points (x_1,y_1) and (x_2,y_2).

- The correlation property of each partial uncertainty is specified by a flag 1 (fully correlated) or 0 (uncorrelated).
- When all partial uncertainties are given in EXFOR, then the heading of the total uncertainty should not be specified in L.6.

When the total uncertainty is known but not all partial uncertainties are known, the "residual uncertainty" (the difference of the quadrature sum of all known partial uncertainties) is calculated for which the user must assign a correlation property.

L6 is used to specify the heading of the total uncertainty, the "residual uncertainty" (the difference of the quadrature sum of all known partial uncertainties) is calculated. The heading of the total uncertainty specified in L.6 must be followed by blanks and one of the following flags:

*: Estimation of the correlation coefficients is skipped (e.g., when the correlation coefficients estimated by the experimentalists are available in EXFOR)

F: The residual uncertainty is fully correlated.

U: The residual uncertainty is uncorrelated.

L7 and below is used for the heading of a partial uncertainty, the heading must be followed by one of the following flags:

0: This partial uncertainty as uncorrelated. (currently not allowed when the total uncertainty heading is flagged with U)

1: This partial uncertainty is fully correlated. (currently not allowed when the total uncertainty heading is flagged with F)

-: This partial uncertainty is subtracted from the total uncertainty. (This is used only when the total uncertainty heading is specified in L.6. The 1st column must be flagged by S.)

Example 4

The partial uncertainty coded under `ERR-S` is specified as uncorrelated. The residual uncertainty is specified as fully correlated.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6: ERR-T       F
7: ERR-S       0
```

Example 5

The partial uncertainty coded under `ERR-1` and `ERR-2` is specified as fully uncorrelated. The residual uncertainty is specified as uncorrelated.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6: ERR-T       U
7: ERR-1       1
8: ERR-2       1
```

Example 6

The partial uncertainty coded under `ERR-S` is specified as uncorrelated, and those under `ERR-1` and `ERR-2` is specified as fully uncorrelated. The residual uncertainty is specified as uncorrelated.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6:
7: ERR-S      0
8: ERR-1      1
9: ERR-2      1
```

Example 7

Same as Example 6, but `ERR-2` is ignored.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6:
7: ERR-S      0
8: ERR-1      1
9:#ERR-2      1
```

Example 8

Same as Example 6, but `MONIT-ERR` instead of `ERR-2`. The flag "S" on the L.9 indicates that the `MONIT-ERR` value is ignored when the dataset is treated as a shape dataset (i.e., MAKTAB is used with the option `-s`).

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6:
7: ERR-S      0
8: ERR-1      1
9:SMONIT-ERR  1
```

Example 9

Same as Example 4, but L.8 indicates that the `MONIT-ERR` is subtracted from the total uncertainty when the dataset is treated as a shape dataset.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6: ERR-T      F
7: ERR-S      0
8:SMONIT-ERR  -
```

Example 10

The partial uncertainty coded under ERR-S is specified as uncorrelated, and a fully correlated partial uncertainty of 1.00% not in EXFOR is added with ERR-A in L.8.

```
1: EN
2: EN
3: EN-ERR
4: DATA
5:
6:
7: ERR-S       0
8: ERR-A       1   1.00 # Normalization uncertainty 1% added
```

Numerical example

Numerical examples are given for the correlation coefficient and total uncertainty Δy calculated with the various setting of correlation flags for a dataset consisting of two data points having the same total and partial uncertainties coded by

```
ERR-T      ERR-S       ERR-1       MONIT-ERR
PER-CENT   PER-CENT    PER-CENT    PER-CENT
 6.         4.          1.          2.
```

for which is the L.6 to L.9 of the HED file are

```
6: ERR-T          ?
7: ERR-S          ?
8: ERR-1          ?
9:SMONIT-ERR      ?
```

The next table summarizes the obtained correlation coefficient between the two data points and the total uncertainty Δy for various combination of the flags. The first column shows use of -s option (treat as a shape dataset) in MAKTAB. "(#)" indicates that the line is commented out.

Shape	ERR-T	ERR-S	ERR-1	MONIT-ERR	Cor	Δy
No	U	1	1	1	0.583	6.000
Yes	U	1	1	1	0.531	5.657
No	F	0	0	(#)	0.528	6.000
No	U	(#)	1	1	0.139	6.000
Yes	U	(#)	1	1	0.031	5.657
No		0	1	1	0.238	4.583
Yes		0	1	1	0.059	4.123
No	F	0	0	-	0.528	6.000
Yes	F	0	0	-	0.469	5.657

Nuclear Data Section
International Atomic Energy Agency
P.O. Box 100
A-1400 Vienna
Austria

e-mail: nds.contact-point@iaea.org
fax: (43-1)26007
telephone: (43-1)2600-21710
web: http://nds.iaea.org/