



INTERNATIONAL ATOMIC ENERGY AGENCY

# NUCLEAR DATA SERVICES

DOCUMENTATION SERIES OF THE IAEA NUCLEAR DATA SECTION

---

## ForEXy: Utility Codes for EXFOR Library

Naohiko Otuka  
IAEA Nuclear Data Section, Vienna, Austria

May 2026

**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 this code package in a publication this should be done in the following way:

N. Otuka, V. Devi, O. Iwamoto “EXFOR utility codes (ForEXy) and their application to neutron fission cross section evaluation”, Applied Radiation and Isotopes **225** (2025) 111903. (<https://doi.org/10.1016/j.apradiso.2025.111903>).

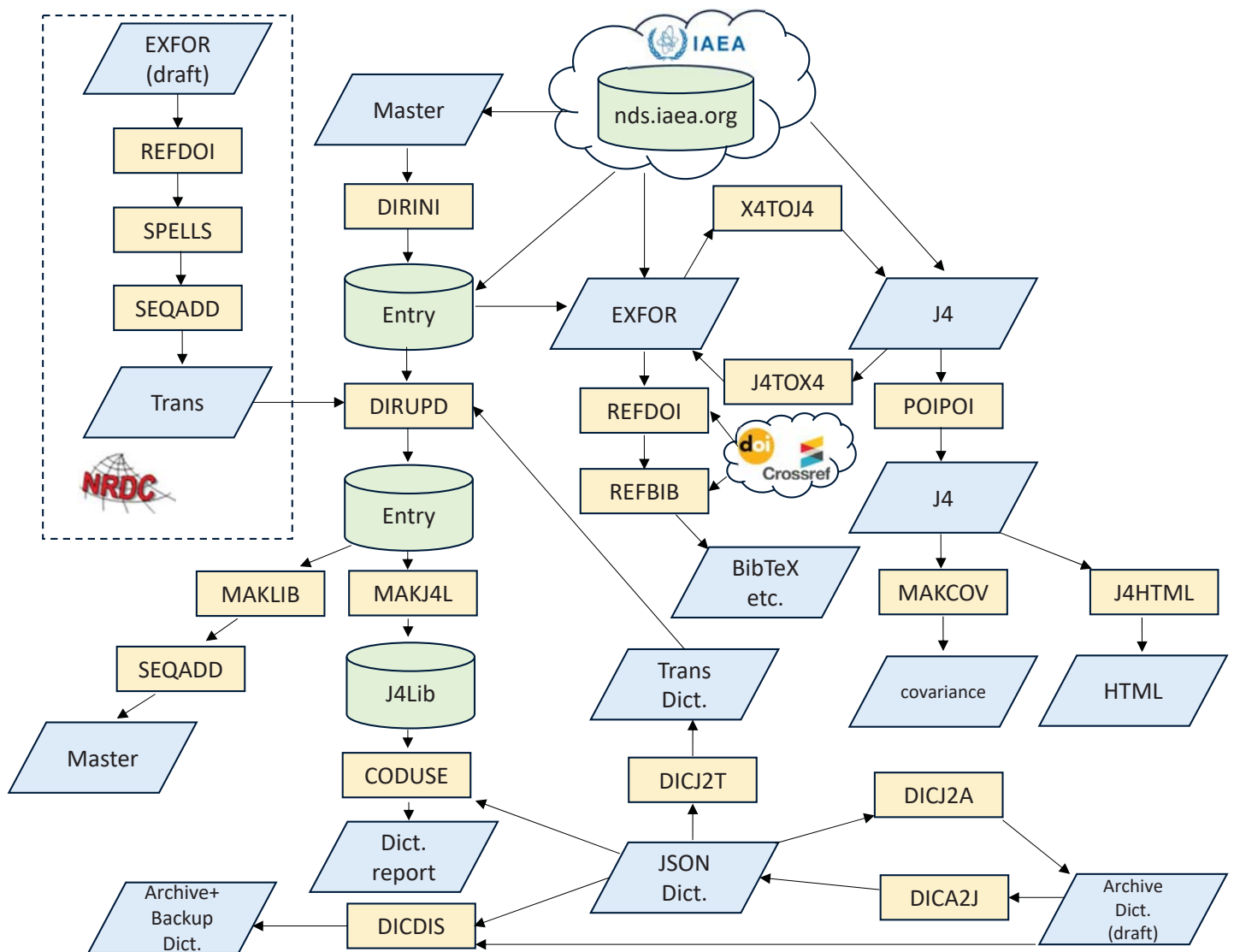
## **ForEXy: Utility Codes for EXFOR Library**

Naohiko Otuka  
IAEA Nuclear Data Section, Vienna, Austria

### **Abstract**

Descriptions are given for a package of utility codes for the files relevant to EXFOR library. This program package is written in Python and may be downloaded from the NRDC software website ([http://nds.iaea.org/nrdc/nrdc\\_sft/](http://nds.iaea.org/nrdc/nrdc_sft/)).

May 2026



## **Contents**

Introduction .....	7
Acknowledgements .....	8
History (major revisions only) .....	8
CODUSE.....	10
DIC227 .....	11
DICA2J.....	12
DICDIS.....	13
DICJ2A.....	14
DICJ2T .....	15
DIRINI .....	16
DIRUPD.....	18
EXTMUL .....	19
J4TOX4 .....	20
J4VIEW .....	21
MAKCOV .....	22
MAKJ4L.....	24
MAKLIB .....	25
POIPOI.....	26
REFBIB .....	28
REFDOI .....	29
SEQADD.....	30
SPELLS .....	31
X4TOJ4 .....	32
X4VIEW.....	34
Appendix 1: Output of formatted free text to HTML by J4VIEW / X4VIEW .....	35
Appendix 2: HED file used in MAKCOV .....	37
Appendix 3: Use a ForEXy code as a module .....	42



## **Introduction**

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

- CODUSE      Analyse use of codes in J4 library.
- 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      Extract 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.
- MAKJ4L      Produce and update J4 library.
- 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.
- REFDI      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/](https://nds.iaea.org/nrdc/nrdc_sft/)) and the PyPI repository (<https://pypi.org/project/forexy/>).

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

### 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 REFDOL.

2025-03-31:

- The package name was renamed from ForEX to ForEXy.
- DIRUPD updated. Now it does not require a log file (e.g., "x4\_dirupd.log") generated in a previous initialization by DIRINI or update by DIRUPD.



2025-12-01:

- First release of CODUSE and MAKJ4L.
- One argument (`keepres0`) was added in the main function of POIPOI.
- The field name “radiation” was changed to “radiation\_type” for the EXFOR keyword RAD-DET in J4.

2026-01-26:

- `-m` was changed to `-n` for controlling of addition of “19” in SEQADD.
- One argument (`email0`) was added in the main function of X4TOJ4.
- The field name “doi” was added to the EXFOR keyword REFERENCE in J4.
- `-e` was changed to `-m` for providing an email address to REFBIB and REFDOI.

2026-05-25:

- First release of J4VIEW and X4VIEW.
- One argument (`file_log`) was added in the main function of X4TOJ4.

## CODUSE

This code reads a J4 file storage (produced by MAKJ4L) and summarizes (1) codes used in an EXFOR entry but not defined in EXFOR/CINDA Dictionary, and (2) codes defined in EXFOR/CINDA Dictionary but not used in EXFOR and CINDA.

### Input files

- J4 files (-i)
- JSON Dictionary (-d)
- CINDA code list (-c)

### Output file

- log file (-o)

### Arguments

- -i *dir\_storage*      directory of input J4 file storage
- -d *file\_dict*        input JSON Dictionary
- -c *file\_cinda*       input list of codes used in CINDA
- -l *file\_log*          output log file [default: "*x4\_coduse.log*"]
- -f      never prompt

### Example

Read J4 files in the J4 file storage *j4*, JSON dictionary *dict\_9131.json* and the CINDA code list *x4\_coduse.cin*, and report the result in a log file *x4\_coduse.log*:

```
python3 x4_coduse.py -i j4 -d dict.json -c x4_coduse.cin
```

## 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<sup>1</sup> (-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 [default: dict\_arc\_new.227]
- -f never prompt

### 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
```

---

<sup>1</sup> 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
- `-f` never prompt

### **Example**

Convert Archive Dictionaries *input/dict\_arc.top*, *input/dict\_arc\_new.001* etc. to a JSON Dictionary *json/dict.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
- `-f` never prompt

### Example

Read Archive and JSON Dictionaries *input/dict\_arc.top*, *input/dict\_arc\_new.001* etc. and *json/dict.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
- -f        never prompt

### **Example**

Convert a JSON Dictionary *json/dict.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 *dict\_ver*                dictionary version (transmission ID)
- -i *dir\_json*                directory of input JSON Dictionary
- -o *dir\_trans*                directory of output TRANS Dictionary
- -f        never prompt

### **Example**

Convert JSON Dictionary *json/dict.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<sup>2</sup>) 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

- directory of entry file storage (-d)
- log file (-g)

### Arguments

- -l *file\_lib* input library tape
- -d *dir\_storage* directory of output entry storage
- -g *file\_log* output log file [default: "x4\_dirupd.log"]
- -f never prompt
- -c delete (1) trailing blanks in col. 12-66, (2) line sequential number (col.67-80) and (3) N2 of ENDBIB/ENDCOMMON/ENDSUBENT/ENDENTRY.

### Example

Initialise the entry storage directory *entry* by loading the input library tape *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 library.txt -d entry
```

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		library.txt

---

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



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/>).

## DIRUPD

This code reads a trans tape<sup>3</sup> (starting from the TRANS record) and adds or updates the entry files in the local storage.

### Input files

- trans tape (-t)
- directory of entry file storage (-d)
- log file (-g)

### Output files

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

### Arguments

- -t *file\_trans* input trans tape
- -d *dir\_storage* (same as DIRINI)
- -g *file\_log* input/output log file [default: "x4\_dirupd.log"]
- -f never prompt
- -c (same as DIRINI. Use this option if you use it for DIRINI.)

### Example

Update of the entry storage *entry* by a tape *trans.txt*, and record it in *x4\_dirupd.log*:

```
python3 x4_dirupd.py -t trans.txt -d entry
```

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		library.txt
1	2023-10-04 00:48:31.725707	1234	20160121	NNDC	trans.txt

---

<sup>3</sup> 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\_inp* 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\_out* output EXFOR file
- -f never prompt

### Example

Extract a dataset 23756.002.3 from an EXFOR file *exfor.txt* to an EXFOR file *exfor\_out.txt* with the dictionary *dict.json*:

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

Extract all datasets from an EXFOR file *exfor.txt* to an EXFOR file *exfor\_out.txt* with the dictionary *dict.json*:

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

Note that the first subentry (e.g., 23756.001) must 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`       input JSON Dictionary
- `-o file_x4`       output EXFOR file
- `-f`       never prompt

### **Example**

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

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

## **J4VIEW**

This code reads a J4 files processed by POIPOI with `-c` and convert it to an HTML file.

### **Input files**

- directory of J4 files processed by POIPOI with `-c` (`-i`)
- JSON Dictionary (`-d`)

### **Output file**

- HTML file (`-o`)

### **Arguments**

- `-i dir_j4` directory of input J4 files processed by POIPOI to read
- `-d file_dict` input JSON Dictionary
- `-c file_css` input CSS file
- `-e entry` EXFOR Entry number
- `-o file_html` output HTML file
- `-l limit` maximum number of data lines to print
- `-g` use sans-serif (gothic) font
- `-u` output with unformatted free text (see **Appendix 1** for free text formatting)

### **Example**

Read J4 files processed by POIPOI with `-c` under the directory *json*, a dictionary *dict.json* and a CSS file *exfor.css*, and write an HTML file *exfor.html* for EXFOR entry E2789:

```
python3 x4_j4view.py -i json -d dict.json -c exfor.css -e E2789 -o exfor.html
```

## MAKCOV

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

### Input files

- J4 file processed by `POIPOI` with `-k all` and without `-p` and `-c (-i)`
- “HED file” (`-j`). See **Appendix 2** of this manual for its details.
- JSON Dictionary (`-d`)

### Output files

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

### Arguments

- `-i file_j4` input J4 file
- `-j file_hed` input “HED file”
- `-d file_dict` input JSON Dictionary
- `-e data_id` EXFOR Dataset ID<sup>4</sup>
- `-o file_cov` output covariance file
- `-g file_log` output log file [default: “*x4\_makcov.log*”]
- `-l x_min` lower boundary of independent variable [default: -9.9E+20]
- `-u x_max` upper boundary of independent variable [default: +9.9E+20]
- `-f` never prompt
- `-s` treat the dataset as a shape (normalization free) dataset
- `-r` print  $\Delta y/y$  (%) rather than  $\Delta y$

---

<sup>4</sup> 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).

### Example

Read a J4 file *exfor.json*, HED file *exfor\_hed.txt*, JSON dictionary *dict.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.json -e  
22742.004.1 -o x4_makcov_out.txt
```

## MAKJ4L

This code reads the entry files in an entry storage and convert them to J4 files in a J4 file storage.

### Input file

- directory of entry file storage (-i)
- JSON Dictionary (-d)

### Output file

- directory of J4 file storage (-j).

### Arguments

- -i *dir\_storage* (same as DIRINI)
- -d *file\_dict* input JSON Dictionary
- -j *dir\_j4lib* directory of output J4 file storage
- -t *file\_trans* trans tape containing entries for J4 file production [default: “none”] (optional)
- -g *file\_log* output log file [default: “x4\_makj4l.log”]
- -m email your email address for addition of DOI to J4 (optional)
- -f never prompt

### Example

Create J4 files under the storage *j4* by converting the entry files in the storage *entry*: with the dictionary *dict.json*:

```
python3 x4_makj4l.py -i entry -d dict.json -j j4
```

Same as above, but do conversion only for entries included in *trans.txt*:

```
python3 x4_makj4l.py -i entry -d dict.json -t trans.txt -j j4
```

Note: When a trans tape is not specified and the specified directory of the J4 file storage exists, all files under the J4 file storage are deleted before starting conversion of the entry files to J4 files.



## **MAKLIB**

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

### Input file

- Directory of entry file storage (-d)

### Output file

- library tape (-l).

### Arguments

- -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)
- -f never prompt
- -a add “19” to two-digit year in N2 of ENTRY/SUBENT/NOSUBENT.
- -c (same as DIRINI.)
- -n exclude dictionaries

### Example

Create a library tape *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 code 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**

- -i *file\_inp* input J4 file
- -d *file\_dict* input JSON Dictionary
- -e *data\_id* EXFOR Dataset ID (or “all” to process all datasets in the input)
- -o *file\_out* output J4 file (or directory if EXFOR Dataset ID is set to “all”)
- -k @*key\_keep* BIB keywords to keep [default: “all”]<sup>5</sup>
- -f never prompt
- -p delete the pointer
- -c keep the common (001) and data subentries separately
- -r keep resonance parameter table<sup>6</sup>

### **Example**

Read a J4 file *exfor.json*, and JSON dictionary *dict.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.json -e 22742.004.1  
-o exfor_poi.json
```

---

<sup>5</sup> The REACTION information is always kept.

<sup>6</sup> With this option, resonance parameters under the pointers are kept in the same subentry without decomposition.

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

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

## **REFBIB**

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 an active internet connection as well as 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**

- -i *ref\_inp* EXFOR reference code or DOI
- -a *fauthor* Family name of the first author [default: “any”]
- -d *file\_dict* input JSON Dictionary
- -o *file\_bib* output bibliography file
- -r *form* output format (doi, piped, exfor, bibtex, json or xml)
- -m *email* your email address (for notification to CrossRef)
- -f never prompt
- -s strip accent symbols in output

### **Example**

For the EXFOR reference code *J,NDS,120,272,2014*, read a JSON dictionary *dict.json* and email address *email@address.com* and write DOI in *x4\_refbib\_out.txt*:

```
python3 x4_refbib.py -i J,NDS,120,272,2014 -d dict.json -o
x4_refbib_out.txt -r doi -m email@address.com
```

## **REFDOI**

This code reads an EXFOR file, and check presence of a DOI for each article coded under REFERENCE, REL-REF, MONIT-REF and STATUS 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 an active internet connection as well as Python modules **pylatexenc**, **pyspellchecker** and **requests**. See the page for REFDOI for their installation.

### **Input files**

- EXFOR file (-i)
- JSON Dictionary (-d)
- Email address (-m)

### **Output file**

- DOI file (-o)

### **Arguments**

- `-i file_x4` input EXFOR file
- `-d file_dict` input JSON Dictionary
- `-o file_doi` output DOI file
- `-j file_json` output JSON file
- `-k @key_anal` keywords for processing [default: "REFERENCE"]
- `-m email` your email address (for notification to CrossRef)
- `-f` never prompt

Note that the DOI file is overwritten while JSON file is updated if exists.

### **Example**

Check reference codes under REFERENCE included in *exfor.txt* with the dictionary *dict.json*:

```
python3 x4_refdoi.py -i exfor.txt -d dict.json -o x4_refdoi_out.txt -j x4_refdoi_out.json -m email@address.com
```

Same as above, analyse codes under REFERENCE and STATUS:

```
python3 x4_refdoi.py -i exfor.txt -d dict.json -o x4_refdoi_out.txt -j x4_refdoi_out.json -k REFERENCE STATUS -m 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

- -i *file\_inp*           input EXFOR file
- -o *file\_out*           output EXFOR file
- -f       never prompt
- -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.
- -d       delete the transmission ID in N6

### 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 them 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\_typo*       output typo list file
- -f       never prompt

### Example

Check spells in an EXFOR entry file *exfor.txt* with a dictionary *x4\_spells.dic* and record the checking result in *x4\_spells\_out.txt*.

```
python3 x4_spells.py -i exfor.txt -d x4_spells.dic -l x4_spells_out.txt
```

## **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**

- `-i file_x4` input EXFOR file
- `-d file_dict` input JSON Dictionary
- `-o file_j4` output J4 file
- `-k @key_keep` BIB keywords to be kept [default: “all”]<sup>7</sup> (optional)
- `-m email` your email address for addition of DOI to J4 (optional)
- `-f` never prompt
- `-c` check record identification of the system identifiers
- `-a` add “19” to two-digit year
- `-g` keep the flag at column 80 of each record of the DATA sections
- `-s` keep real numbers as strings

Note that this code sets the integers in the `BIB(N2)`, `ENDBIB(N1)`, `COMMON(N2)`, `ENDCOMMON(N1)`, `ENDDATA(N1)` and `ENDSUBENT (N1)` of the J4 file to 0.

### **Example**

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

```
python3 x4_x4toj4.py -i exfor.txt -d dict.json -o exfor.json -k all
```

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

---

<sup>7</sup> The `REACTION` information is always kept.



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

## X4VIEW

This code reads an EXFOR file, process it by SEQADD, and convert it to an HTML file. Three other Python scripts files for J4VIEW, POIPOI, SEQADD and X4TOJ4 must be placed together with X4VIEW in the same directory.

### Input files

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

### Output file

- HTML file (-o)

### Arguments

- `-i file_x4` input EXFOR file
- `-d file_dict` input JSON Dictionary
- `-c file_css` input CSS file
- `-e entry` EXFOR Entry number
- `-o file_html` output HTML file
- `-l limit` maximum number of data lines to print
- `-m email` your email address for addition of DOI (optional)
- `-f` never prompt
- `-g` use sans-serif (gothic) font
- `-u` output with unformatted free text (see **Appendix 1** for free text formatting)

### Example

Read an EXFOR file *exfor.txt*, JSON dictionary *dict.json* and a CSS file *exfor.css*, and write an HTML file *exfor.html* for EXFOR entry E2789:

```
python3 x4_x4view.py -i exfor.txt -d dict.json -c exfor.css -e E2789 -o exfor.html
```

## **Appendix 1: Output of formatted free text to HTML by J4VIEW / X4VIEW**

By running J4VIEW or X4VIEW without the unformatted free text output option (-u), one can introduce some structures to the free text field of the input EXFOR entry. Currently, paragraph, heading, itemizing and table are supported.

### 1. Paragraph

Presence of an alphabetical character or number at column 12 indicates opening of a new paragraph, and the free text is indented.

### 2. Heading

Presence of an asterisk at column 12 indicates a heading line, and the free text is not indented.

### 3. Itemizing

Presence of a minus (-) or full stop (.) character at column 12 followed by a blank indicates an item, and the free text follows a bullet.

### 4. Table

Presence of a plus character followed by two minus character (+--) indicates beginning or ending of a table in free text.

*Example:*

#### **Input**

SAMPLE	Metallic foils of Pt, Ti and Al used:	E281100100011
	+-----+	E281100100012
	Pt Ti Al	E281100100013
	-----	E281100100014
	Nominal purity(%) 99.95 99.6 >99	E281100100015
	-----	E281100100016
	Thickness (mg/cm2) 21.672 2.343 1.817	E281100100017
	+-----+	E281100100018
METHOD	Foils were	E281100100019
	- irradiated for 52 min at ~204 nA	E281100100020
	- cooled for 54 min	E281100100021
COMMENT	* By Naohiko Otuka (2025-08-10):	E281100100022
	This work was carried out at RI Beam Factory operated	E281100100023
	by RIKEN Nishina Center and CNS, University of Tokyo,	E281100100024
	Japan.	E281100100025
	The authors would like to thank Dr. Y. Shigekawa, Dr.	E281100100026
	Y. Kanayama, and Dr. H. Shimizu (RIKEN Nishina Center)	E281100100027
	for technical assistance with the experiment.	E281100100028

## Output

### **Sample**

Metallic foils of Pt, Ti and Al used:

	Pt	Ti	Al
Nominal purity (%)	99.95	99.6	>99
Thickness (mg/cm <sup>2</sup> )	21.672	2.343	1.817

### **Method**

Foils were

- irradiated for 52 min at ~204 nA
- cooled for 54 min

### **Comment**

\* By Naohiko Otuka (2025-08-10):

This work was carried out at RI Beam Factory operated by RIKEN Nishina Center and CNS, University of Tokyo, Japan.

The authors would like to thank Dr. Y. Shigekawa, Dr. Y. Kanayama, and Dr. H. Shimizu (RIKEN Nishina Center) for technical assistance with the experiment.

## **Appendix 2: HED file used in MAKCOV**

(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 MAKCOV. 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 (MAKCOV 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,  $\Delta x$ , y,  $\Delta y$ ) 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 “-“.  $\Delta y$  is calculated by adding these partial uncertainties in % quadratically.

#### Example 1

$\Delta x$  is calculated by the difference between EN-MIN and EN-MAX.  $\Delta y$  is taken from ERR-T.

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

#### Example 2

$\Delta x$  is taken from EN-ERR.  $\Delta y$  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

$\Delta x$  is always zero.  $\Delta y$  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., MAKCOV 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., `MAKCOV` 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  $\Delta y$  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 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  $\Delta y$  for various combination of the flags. The first column shows use of -s option (treat as a shape dataset) in MAKCOV. “( # )” indicates that the line is commented out.

Shape	ERR-T	ERR-S	ERR-1	MONIT-ERR	Cor	$\Delta y$
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

### **Appendix 3: Use a ForEXy code as a module**

The ForEXy package is available on the PyPI repository (<https://pypi.org/project/forexy/>), and can be installed by

```
pip install forexy
```

Then one can load the main function of each code as an external library. See “Arguments” of the each code description for the list of the arguments. The module must be called with all arguments in the same order.

An argument name starting from @ indicate that it must be given as a list. (e.g., @key\_keep of POIPOI and X4TOJ4 and @key\_anal of REFDOI).

#### **Example 1**

The following script (utilizing the ForEXy code REFBIB) reads an EXFOR reference code or DOI, and output a BibTeX entry:

```
from forexy import x4_refbib

def main():

    ref_inp=input("Input EXFOR reference code or DOI -> ")
    fauthor   = "any"
    file_dict = "dict.json"
    file_bib  = "exfor.bib"
    format    = "bibtex"
    email     = "email@address.com"
    force     = False
    strip     = False

    x4_refbib.main\
(ref_inp,fauthor,file_dict,file_bib,format,email,force,strip)

    f=open(file_bib, 'r')
    lines=f.readlines()
    f.close

    for line in lines:
        print(line, end="")

    return

if __name__ == "__main__":
    main()
    exit()
```

### Example 2

The following script (utilizing the ForEXy code X4TOJ4) reads an EXFOR file, and output a J4 file for the keywords AUTHOR, REFERENCE, FACILITY and REACTION:

```
from forexy import x4_x4toj4

def main():

    file_x4    =input("Input EXFOR file name -> ")
    file_dict  = "dict.json"
    file_j4    = "exfor.json"
    key_keep   = ["AUTHOR", "REFERENCE", "FACILITY", "REACTION"]
    force      = "False"
    check      = "True"
    add19      = "True"
    flag       = "False"
    string     = "False"

    x4_x4toj4.main\
    (file_x4,file_dict,file_j4,key_keep,force,check,add19,flag,string)

    f=open(file_j4, 'r')
    lines=f.readlines()
    f.close

    for line in lines:
        print(line, end="")

    return

if __name__ == "__main__":
    main()
    exit()
```





---

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

---

e-mail: [nds.contact-point@iaea.org](mailto:nds.contact-point@iaea.org)  
fax: (43-1)26007  
telephone: (43-1)2600-21710  
web: <https://nds.iaea.org/>