JSON-Tree Editor

by Viktor Zerkin

~ independent software developer ~

EXFOR Workshop: Compilation of Experimental Nuclear Reaction Data
3 - 6 December 2024, IAEA Headquarters, Vienna, Austria

Part I.

Introduction. Purpose. Concept.

Introduction. JSON.

JSON (JavaScript Object Notation) is a lightweight data-interchange format

JSON is based on a subset of the JavaScript since 1999. JSON is a text format that is language independent but uses conventions familiar to programmers of the C-family of languages (C/C++, C#, Java, JavaScript), Perl, Python, and many others.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

Advantages

- vs. plain text: easy to describe complex data and nested structures; many programming languages provide Parser (easier for programming of read/write operations)
- vs. XML: simpler, much shorter (~ ×5 times)
- vs. DB: do not need special tools to view data (files are just text)
- super-easy to manipulate in programs in JavaScript (Web) and Python (desktop)

Disadvantages

- vs. plain text: extra-text (*key*, : and “ “).

  Example of parameter and named parameter in plain text and JSON:

  text file: `A1234` *#program uses assigned variable name, known location in the file and format*

  text file: `ENTRY A1234` *#program uses known location in the file and format; name changed dynamically*

  json file: `"ENTRY":"A1234"` *#program needs separators: +5 characters (with flexible location)*
- vs. XML: less flexible (XML: *<tag attributes>body</tag>*, JSON: *{key:value}* no *attributes*)
- vs. DB: need extra tools for data search and manipulations
- vs. JavaScript/object and Python/dict: no comments allowed, double-quats

  alternative: “JSON5 – JSON for Humans” - see https://json5.org/

JSON for nuclear data.

JSON for Nuclear Data

Web retrieval systems *(examples on my own experience)*

- JavaScript programming of dynamic pages: update page asynchronously without full reloading using JSON in communication with Web-Server via AJAX (Asynchronous JavaScript and XML).
  Example: EE-View.
- View page with retrieved data via JSON: simpler Web-programming, re-use JSON files for Web-API for external users with remote programs.
  2000. Web-pages: DB => Java => html+JavaScript => JS-code => Internet => html in Web-browser
  2020. Web-pages: DB => Java => html+JSON => JS-code => Internet => html in Web-browser
  + Web-API: DB => Java => JSON => Internet => User's program/Python/Java/wget/curl/etc.
  Examples: ENDF Decay-Data view and comparison; Web-API for EXFOR, ENDF, IBANDL

Data and formats modernization, new technologies, new tasks, new developers, new users

(related not only to new IT technologies and the younger generation, but also to a more dynamic rotation policy)

- Modernize and develop new formats in data centers and user communities to improve/simplify data access, increase data usage, making data accessible to more users, reducing the barrier to entrance.
  Examples: NRDC (X5), NDS (J4, Dictionary), NNDC (ENSDF), WPEC-SG50 (MEDUSAL)
- NRDC-2024: A71 All (Continuing action) Analyze X5 structure/hierarchy and contents, contact Zerkin with questions and proposals.
- Off-line distribution to "large users" using entire content of nuclear data libraries to build ML and other systems based on new technologies.

Structure, options and agreement on JSON formats for nuclear data

(Note. JSON is a notation that can be used to define a format, but not the format itself)

- Although it is easy to produce data in JSON as output from modern programs, the format (schema/structure and data types) delivered to end-users should be correct (cross-checked) and stable (agreed between data centers and fixed for observable time). We need to understand proposed JSON formats and discuss options.
- To achieve this, we need a JSON-Viewer, and possibly, an Editor, and preferably Web based.

Before development. Development loop

Steps to go.

1. Understand problem and possibly task(s)
2. Three questions to answer:
   a) What already exists and can be used?
   b) What is wrong now?
   c) What do we want to achieve? (define ideal goal)
3. Preparations:
   a) discuss/define main ideas, concept and possible technologies
   b) study and test technologies, select technology
4. Plan: split problem to tasks/sub-tasks, define dates
5. Implementation: development loop
   a) Define/invent data structures and algorithms
   b) Developing program/modules + testing + feedback and correction tasks
   c) Go to -2 = 5.a) or to 4. or even to 3. when needed

Our needs and existing JSON web-tools

Our purposes

- understand proposed JSON based nuclear data formats on examples (!!!)
- test JSON files (investigate structure, evaluate rationality), find mistakes, report bugs
- discuss and modify JSON files to make counterproposals and improve formats
- accept proposed JSON formats and make sure that it is common agreed format

Some Web links used before development

JSON tree, XML, beautifiers:

https://countwordsfree.com/jsonviewer

https://codebeautify.org/jsonviewer

example with URL:

https://codebeautify.org/jsonviewer?https://codebeautify.org/jsonviewer?url=https://nds.iaea.org/exfor/x4guide/x5json/23114002.x4z.txt

JSON validators: https://jsonlint.com/ https://codebeautify.org/jsonvalidator

JSON-schema: https://www.nndc.bnl.gov/ensdfschema/

Properties (why insufficient for our purposes)

- general purpose products (no specialized features)
- work only via Web (no local version)
- fixed functionality (no user's extensions and new functions)

JSON-Tree Editor: a tool to understand and discuss nuclear data formats

https://vzerkin.github.io/
https://zerkin.usite.pro/edit-json-tree/

For the moment, we can discuss JSON of EXFOR (~5 versions), Dictionaries (3 versions), ENDF-MF3/33, 4/34, Decay-data; IBANDL; ENSDF; NSR-output

Links with preload examples:

https://zerkin.usite.pro/edit-json-tree/	example: start with types of JSON objects
https://zerkin.usite.pro/edit-json-tree/#0	new: start with empty JSON
https://zerkin.usite.pro/edit-json-tree/#1	X5json for EXFOR #13597
https://zerkin.usite.pro/edit-json-tree/#4	Dictionary by N.Otsuka (9130)
https://zerkin.usite.pro/edit-json-tree/#4o	Dictionary by S.Okumura
https://zerkin.usite.pro/edit-json-tree/#4z	Dictionary for Apps by V.Zerkin
https://zerkin.usite.pro/edit-json-tree/#5	ensdf-json by NNDC
https://zerkin.usite.pro/edit-json-tree/#7	nsr_result by NNDC
https://zerkin.usite.pro/edit-json-tree/#9	exfor_json by S.Okumura
https://zerkin.usite.pro/edit-json-tree/#10	EXFOR-Std in JSON by V.Zerkin (2019)
https://zerkin.usite.pro/edit-json-tree/cmp2h.htm	parallel: compare 2 JSON files (horizontal)
https://zerkin.usite.pro/edit-json-tree/cmp2v.htm	parallel: compare 2 JSON files (vertical)
https://zerkin.usite.pro/edit-json-tree/cmp3vh.htm	parallel: compare 2 JSON files (vert.+hor.)
https://zerkin.usite.pro/edit-json-tree/cmp3dict.htm	compare 3 Dictionaries: SO | NO | ZV

Concept and technology

JSON-Tree Editor

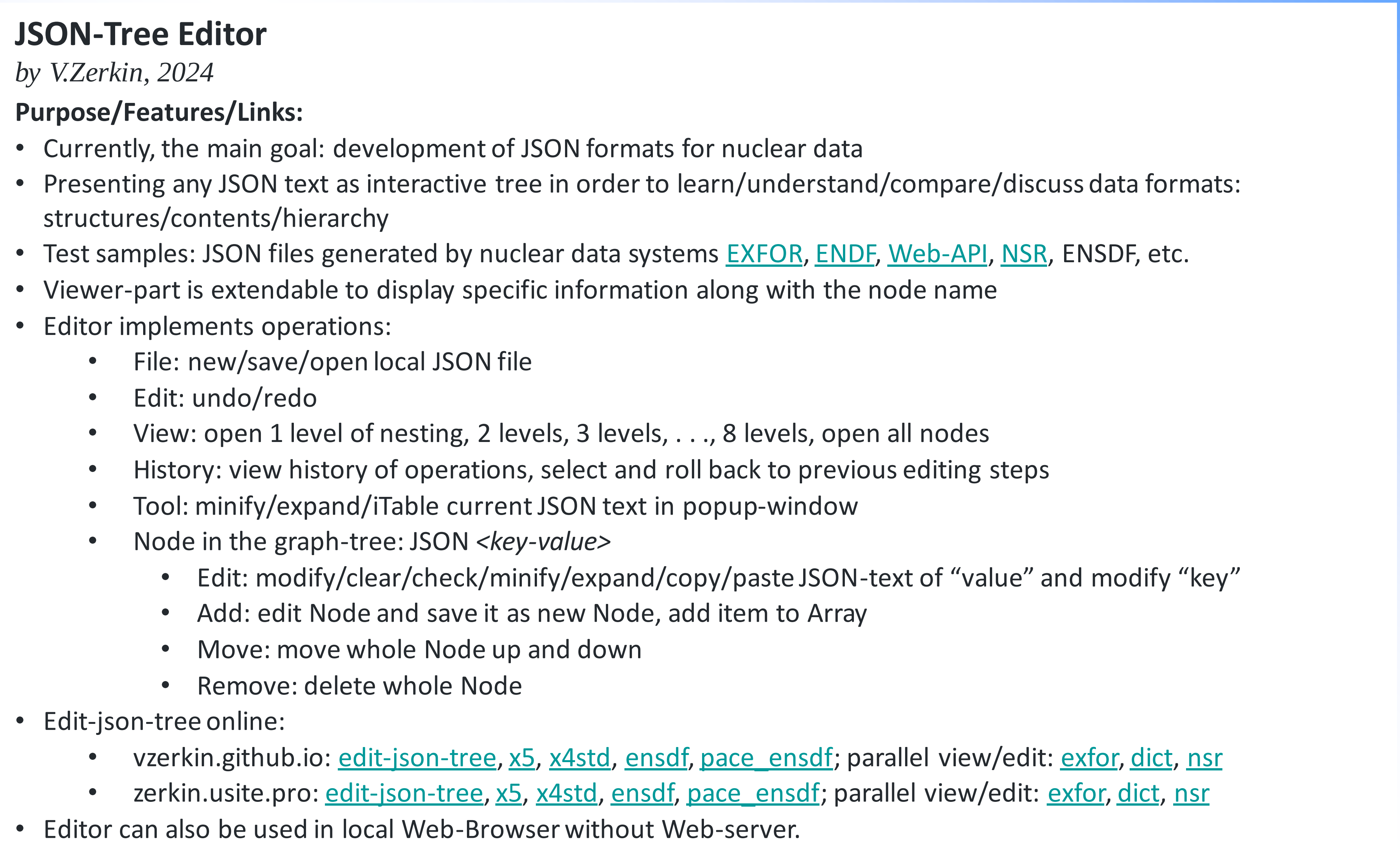
1. presents JSON as interactive-tree
2. is an Application running in Web-browser
    a) written in JavaScript
    b) work from Web-server and locally (no server needed)
    c) platform independent (running “inside” Web browser on any OS)
    d) input: copy/paste or select local JSON file
    e) output: save JSON to browser’s download area
3. looks and operates like “native” App - intuitive for users
4. implements traditional editor’s functionality
5. provides extra-fuctionality for known nuclear data formats
6. is easy for extensions

My past experience with Web-iTree and Web-editors:

x4±, ensdf±, web-editors: exfor (draft), ensdf

Front-page on GitHub. Overview.

JSON-Tree Editor: https://vzerkin.github.io/



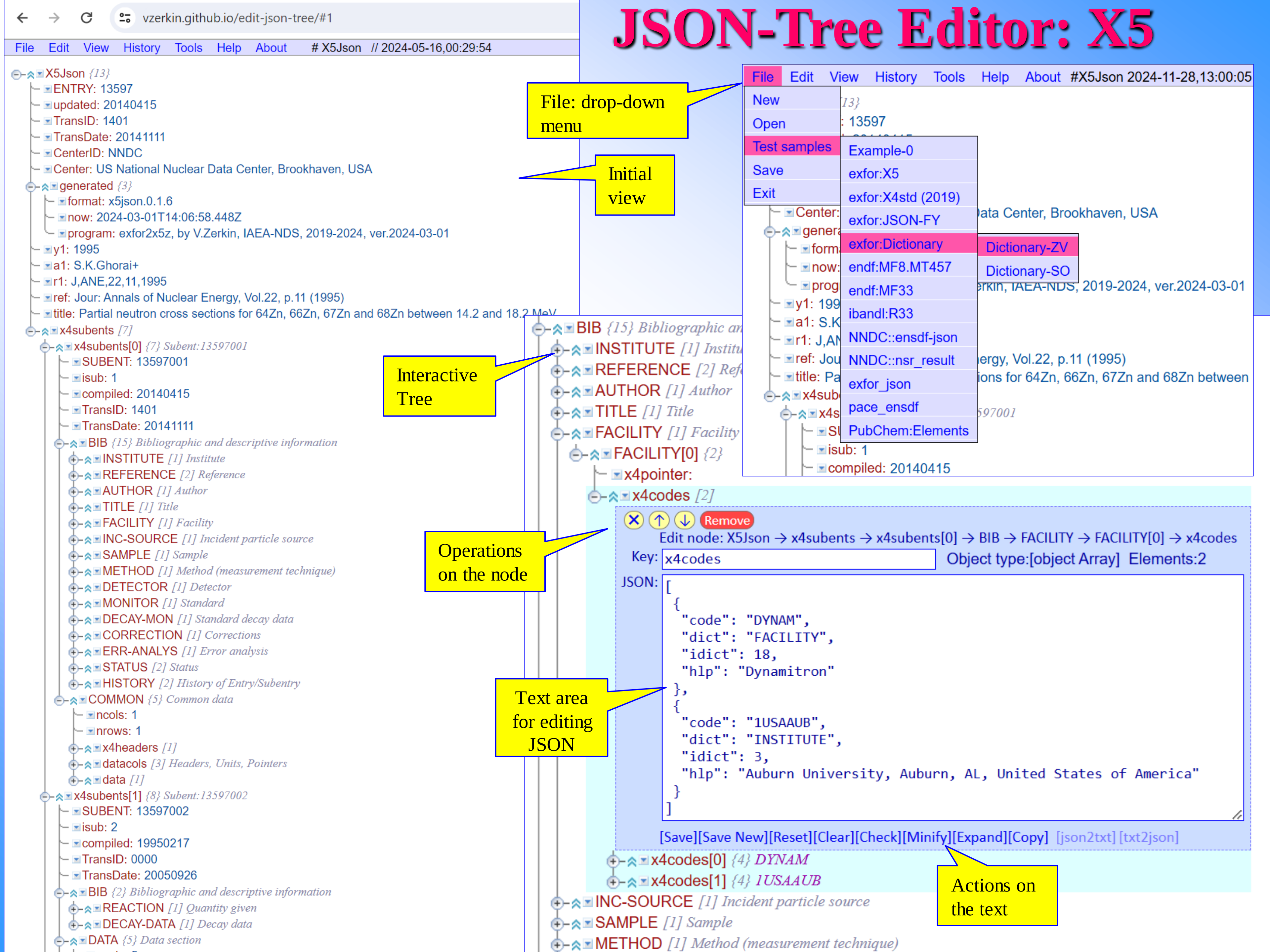
JSON-Tree Editor

by V.Zerkin, 2024

Purpose/Features/Links:

- Currently, the main goal: development of JSON formats for nuclear data
- Presenting any JSON text as interactive tree in order to learn/understand/compare/discuss data formats: structures/contents/hierarchy
- Test samples: JSON files generated by nuclear data systems EXFOR, ENDF, Web-API, NSR, ENSDF, etc.
- Viewer-part is extendable to display specific information along with the node name
- Editor implements operations:
  - File: new/save/open local JSON file
  - Edit: undo/redo
  - View: open 1 level of nesting, 2 levels, 3 levels, . . ., 8 levels, open all nodes
  - History: view history of operations, select and roll back to previous editing steps
  - Tool: minify/expand/iTable current JSON text in popup-window
  - Node in the graph-tree: JSON <*key-value*>
    - Edit: modify/clear/check/minify/expand/copy/paste JSON-text of “value” and modify “key”
    - Add: edit Node and save it as new Node, add item to Array
    - Move: move whole Node up and down
    - Remove: delete whole Node
- Edit-json-tree online:
  - vzerkin.github.io: edit-json-tree, x5, x4std, ensdf, pace_ensdf; parallel view/edit: exfor, dict, nsr
  - zerkin.usite.pro: edit-json-tree, x5, x4std, ensdf, pace_ensdf; parallel view/edit: exfor, dict, nsr
- Editor can also be used in local Web-Browser without Web-server.

JSON-Tree Editor: X5



File: drop-down menu

Initial view

Interactive Tree

Operations on the node

Text area for editing JSON

Actions on the text

```
vzerkin.github.io/edit-json-tree/#1
File  Edit  View  History  Tools  Help  About    # X5Json  // 2024-05-16,00:29:54

X5Json {13}
  ENTRY: 13597
  updated: 20140415
  TransID: 1401
  TransDate: 20141111
  CenterID: NNDC
  Center: US National Nuclear Data Center, Brookhaven, USA
  generated {3}
    format: x5json.0.1.6
    now: 2024-03-01T14:06:58.448Z
    program: exfor2x5z, by V.Zerkin, IAEA-NDS, 2019-2024, ver.2024-03-01
  y1: 1995
  a1: S.K.Ghorai+
  r1: J,ANE,22,11,1995
  ref: Jour: Annals of Nuclear Energy, Vol.22, p.11 (1995)
  title: Partial neutron cross sections for 64Zn, 66Zn, 67Zn and 68Zn between 14.2 and 18.2 MeV
  x4subents [7]
    x4subents[0] {7} Subent:13597001
      SUBENT: 13597001
      isub: 1
      compiled: 20140415
      TransID: 1401
      TransDate: 20141111
      BIB {15} Bibliographic and descriptive information
        INSTITUTE [1] Institute
        REFERENCE [2] Reference
        AUTHOR [1] Author
        TITLE [1] Title
        FACILITY [1] Facility
        INC-SOURCE [1] Incident particle source
        SAMPLE [1] Sample
        METHOD [1] Method (measurement technique)
        DETECTOR [1] Detector
        MONITOR [1] Standard
        DECAY-MON [1] Standard decay data
        CORRECTION [1] Corrections
        ERR-ANALYS [1] Error analysis
        STATUS [2] Status
        HISTORY [2] History of Entry/Subentry
      COMMON {5} Common data
        ncols: 1
        nrows: 1
        x4headers [1]
        datacols [3] Headers, Units, Pointers
        data [1]
    x4subents[1] {8} Subent:13597002
      SUBENT: 13597002
      isub: 2
      compiled: 19950217
      TransID: 0000
      TransDate: 20050926
      BIB {2} Bibliographic and descriptive information
        REACTION [1] Quantity given
        DECAY-DATA [1] Decay data
      DATA {5} Data section
```

```
File  Edit  View  History  Tools  Help  About  #X5Json 2024-11-28,13:00:05

New
Open
Test samples
    Example-0
    exfor:X5
    exfor:X4std (2019)
    exfor:JSON-FY
    exfor:Dictionary
        Dictionary-ZV
        Dictionary-SO
    endf:MF8.MT457
    endf:MF33
    ibandl:R33
    NNDC::ensdf-json
    NNDC::nsr_result
    exfor_json
    pace_ensdf
    PubChem:Elements
Save
Exit
```

```
BIB {15} Bibliographic an
  INSTITUTE [1] Institu
  REFERENCE [2] Ref
  AUTHOR [1] Author
  TITLE [1] Title
  FACILITY [1] Facility
    FACILITY[0] {2}
      x4pointer:
      x4codes [2]
```

ⓧ ⓘ ⓘ Remove

Edit node: X5Json → x4subents → x4subents[0] → BIB → FACILITY → FACILITY[0] → x4codes

Key: x4codes   Object type:[object Array] Elements:2

JSON:

```
[
 {
  "code": "DYNAM",
  "dict": "FACILITY",
  "idict": 18,
  "hlp": "Dynamitron"
 },
 {
  "code": "1USAAUB",
  "dict": "INSTITUTE",
  "idict": 3,
  "hlp": "Auburn University, Auburn, AL, United States of America"
 }
]
```

[Save][Save New][Reset][Clear][Check][Minify][Expand][Copy] [json2txt] [txt2json]

```
      x4codes[0] {4} DYNAM
      x4codes[1] {4} 1USAAUB
  INC-SOURCE [1] Incident particle source
  SAMPLE [1] Sample
  METHOD [1] Method (measurement technique)
```

Part II.

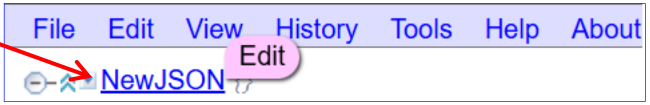
Start. Input. Viewer.

Start. Input JSON.

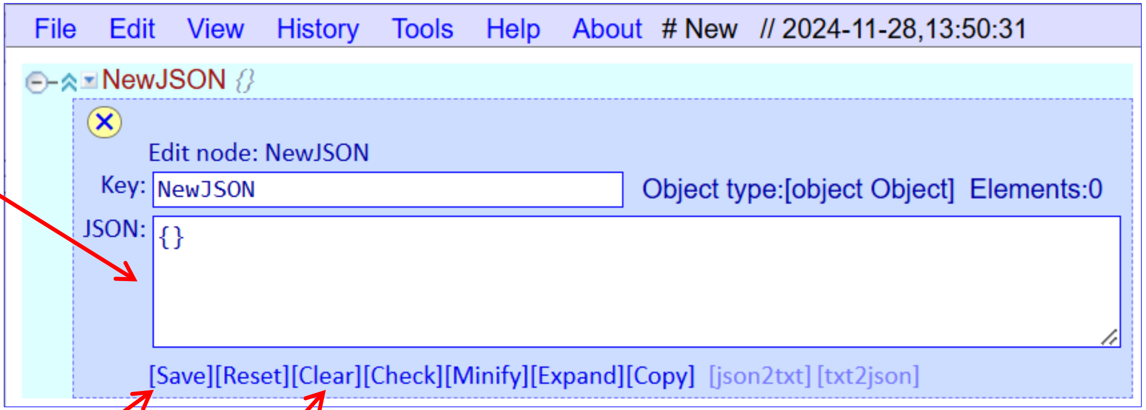
Start. Open in Web-Browser URL address: https://zerkin.usite.pro/edit-json-tree/

Input.

1) File → Open → Select file on you PC
2) File → New   #start with empty JSON object: {}
3) File → Test samples → Select example of JSON
4) Copy/Paste:
    1) Mouse-over any Node-Key displays tool-tip prompt: [Edit]
    2) Click on Node-Key



    3) It will open editing-area



New // 2024-11-28,13:50:31

Edit node: NewJSON

Key: NewJSON Object type:[object Object] Elements:0

JSON: {}

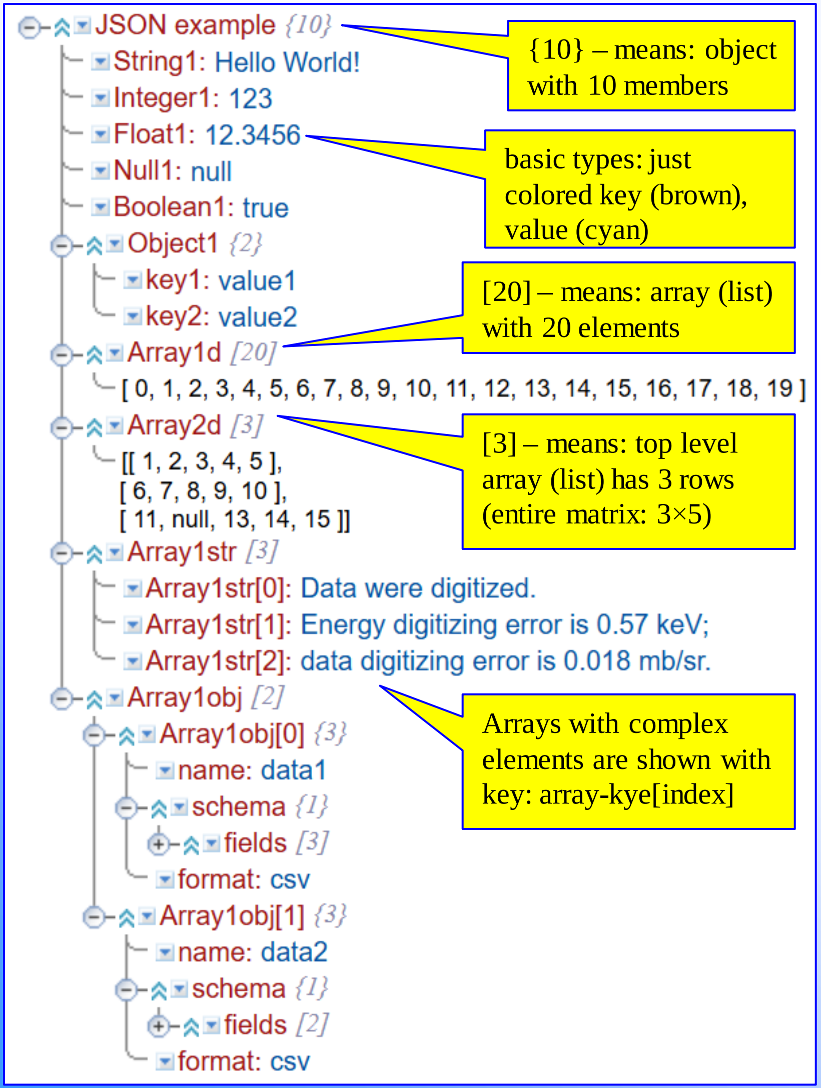
    4) Use operations: [Clear] and keyboard: <Ctrl/V> to paste text.

Note. <Ctrl/A>,<Ctrl/C>,<Ctrl/V>,<Ctrl/Z> work on text as usual

    5) Use operation [Save] to store JSON as Node in the tree

JSON-Viewer

By default JSON data are presented by pairs <key:value> or <key:object> in the tree graph with colors and display object type and length.



```
JSON example {10}
  String1: Hello World!
  Integer1: 123
  Float1: 12.3456
  Null1: null
  Boolean1: true
  Object1 {2}
    key1: value1
    key2: value2
  Array1d [20]
    [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 ]
  Array2d [3]
    [[ 1, 2, 3, 4, 5 ],
    [ 6, 7, 8, 9, 10 ],
    [ 11, null, 13, 14, 15 ]]
  Array1str [3]
    Array1str[0]: Data were digitized.
    Array1str[1]: Energy digitizing error is 0.57 keV;
    Array1str[2]: data digitizing error is 0.018 mb/sr.
  Array1obj [2]
    Array1obj[0] {3}
      name: data1
      schema {1}
        fields [3]
      format: csv
    Array1obj[1] {3}
      name: data2
      schema {1}
        fields [2]
      format: csv
```

{10} – means: object with 10 members

basic types: just colored key (brown), value (cyan)

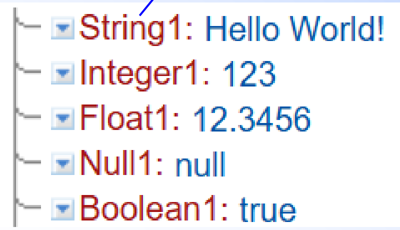
[20] – means: array (list) with 20 elements

[3] – means: top level array (list) has 3 rows (entire matrix: 3×5)

Arrays with complex elements are shown with key: array-kye[index]

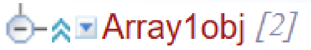
Simple values without further structuring

Click on Key → open edit panel
(second click → close edit panel)



```
String1: Hello World!
Integer1: 123
Float1: 12.3456
Null1: null
Boolean1: true
```

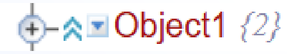
Array [length]



```
Array1obj [2]
```

Collapse (open)

Object [length]

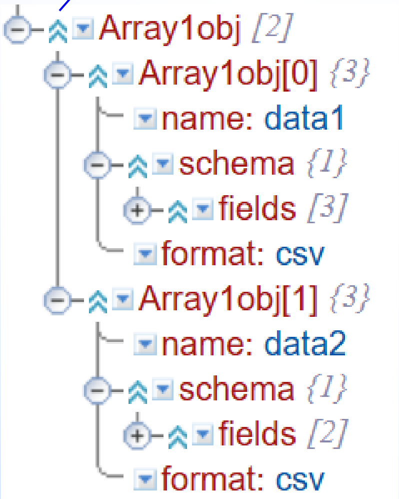


```
Object1 {2}
```

Open (collapse)

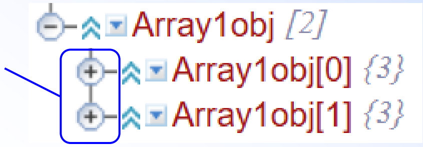
JSON-Viewer. Extended functionality.

Pull up the lower level *(no need to collapse very node below)*



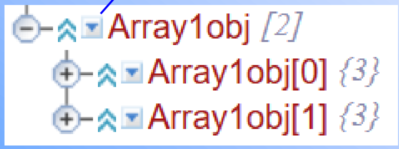
```
Array1obj [2]
  Array1obj[0] {3}
    name: data1
    schema {1}
      fields [3]
    format: csv
  Array1obj[1] {3}
    name: data2
    schema {1}
      fields [2]
    format: csv
```

Collapsed

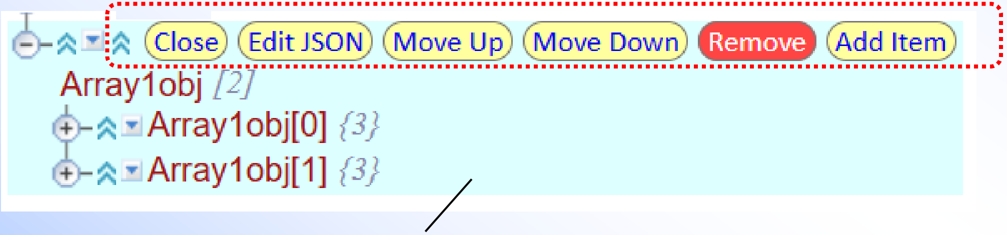


```
Array1obj [2]
  Array1obj[0] {3}
  Array1obj[1] {3}
```

Open navigation panel



```
Array1obj [2]
  Array1obj[0] {3}
  Array1obj[1] {3}
```



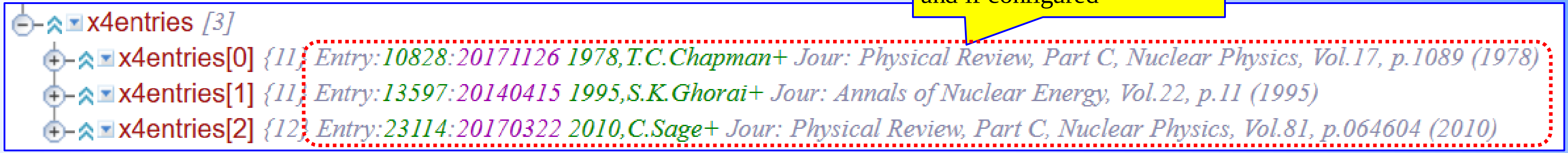
```
Array1obj [2]
  Array1obj[0] {3}
  Array1obj[1] {3}
```

Note. Cyan background outlines content of affected node

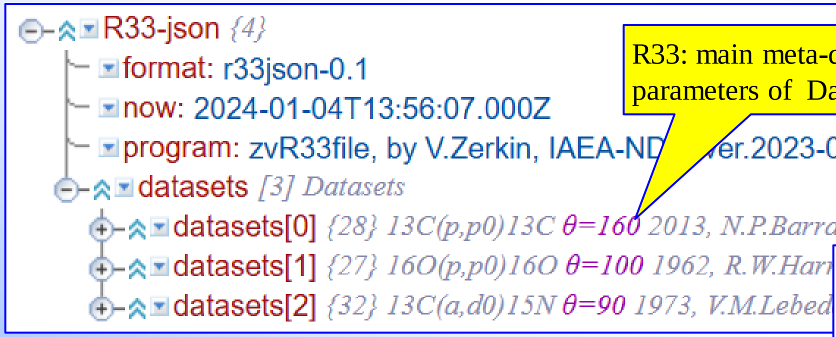
JSON-Viewer

For known JSON formats, data can be displayed with additional useful information.
The idea is to configure viewer as necessary.

Additional info when useful and if configured

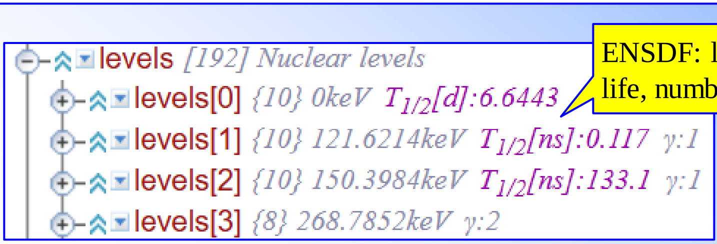


x4entries *[3]*
x4entries[0] *{11} Entry:10828:20171126 1978,T.C.Chapman+ Jour: Physical Review, Part C, Nuclear Physics, Vol.17, p.1089 (1978)*
x4entries[1] *{11} Entry:13597:20140415 1995,S.K.Ghorai+ Jour: Annals of Nuclear Energy, Vol.22, p.11 (1995)*
x4entries[2] *{12} Entry:23114:20170322 2010,C.Sage+ Jour: Physical Review, Part C, Nuclear Physics, Vol.81, p.064604 (2010)*



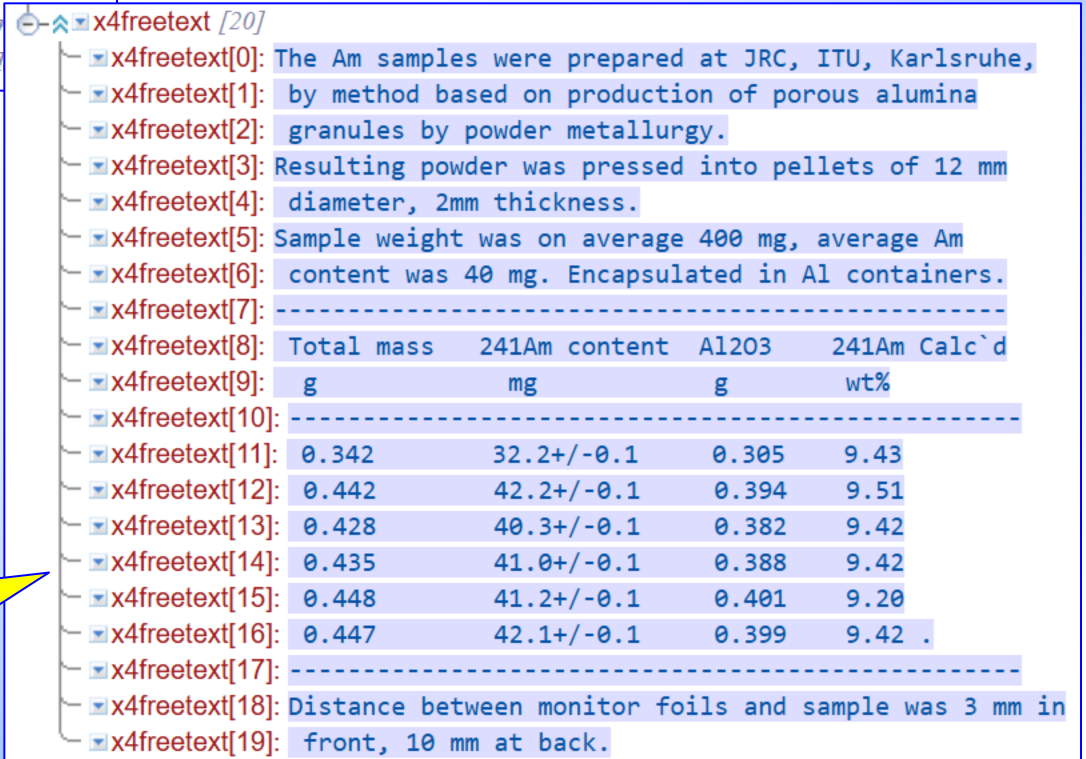
R33-json *{4}*
format: r33json-0.1
now: 2024-01-04T13:56:07.000Z
program: zvR33file, by V.Zerkin, IAEA-ND ver.2023-02-09
datasets *[3] Datasets*
datasets[0] *{28} 13C(p,p0)13C θ=160 2013, N.P.Barradas*
datasets[1] *{27} 16O(p,p0)16O θ=100 1962, R.W.Harr*
datasets[2] *{32} 13C(a,d0)15N θ=90 1973, V.M.Lebed*

R33: main meta-data and parameters of Datasets



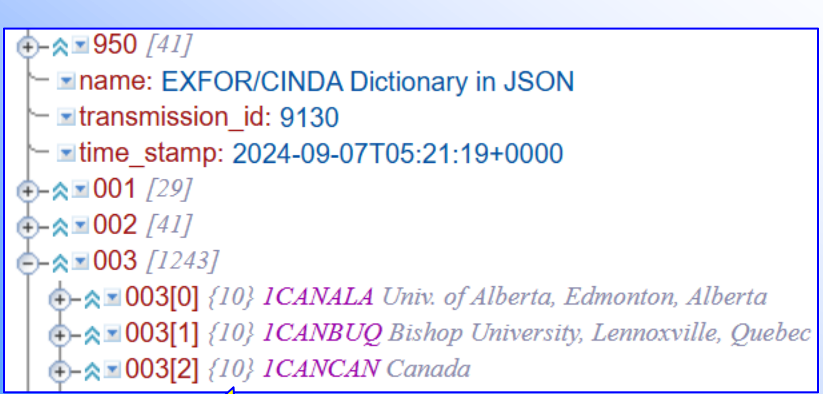
levels *[192] Nuclear levels*
levels[0] *{10} 0keV $T_{1/2}$[d]:6.6443*
levels[1] *{10} 121.6214keV $T_{1/2}$[ns]:0.117 γ:1*
levels[2] *{10} 150.3984keV $T_{1/2}$[ns]:133.1 γ:1*
levels[3] *{8} 268.7852keV γ:2*

ENSDF: level energy, half-life, number of gammas



x4freetext *[20]*

```
x4freetext[0]: The Am samples were prepared at JRC, ITU, Karlsruhe,
x4freetext[1]:  by method based on production of porous alumina
x4freetext[2]:  granules by powder metallurgy.
x4freetext[3]: Resulting powder was pressed into pellets of 12 mm
x4freetext[4]:  diameter, 2mm thickness.
x4freetext[5]: Sample weight was on average 400 mg, average Am
x4freetext[6]:  content was 40 mg. Encapsulated in Al containers.
x4freetext[7]: --------------------------------------------------
x4freetext[8]:  Total mass    241Am content   Al2O3    241Am Calc`d
x4freetext[9]:   g               mg             g         wt%
x4freetext[10]: --------------------------------------------------
x4freetext[11]:  0.342        32.2+/-0.1     0.305     9.43
x4freetext[12]:  0.442        42.2+/-0.1     0.394     9.51
x4freetext[13]:  0.428        40.3+/-0.1     0.382     9.42
x4freetext[14]:  0.435        41.0+/-0.1     0.388     9.42
x4freetext[15]:  0.448        41.2+/-0.1     0.401     9.20
x4freetext[16]:  0.447        42.1+/-0.1     0.399     9.42 .
x4freetext[17]: --------------------------------------------------
x4freetext[18]: Distance between monitor foils and sample was 3 mm in
x4freetext[19]:  front, 10 mm at back.
```



950 *[41]*
name: EXFOR/CINDA Dictionary in JSON
transmission_id: 9130
time_stamp: 2024-09-07T05:21:19+0000
001 *[29]*
002 *[41]*
003 *[1243]*
003[0] *{10} 1CANALA Univ. of Alberta, Edmonton, Alberta*
003[1] *{10} 1CANBUQ Bishop University, Lennoxville, Quebec*
003[2] *{10} 1CANCAN Canada*

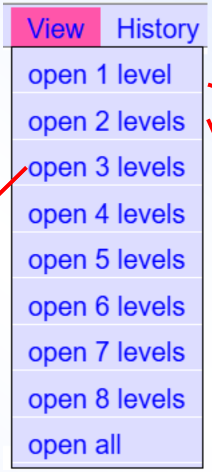
Dictionary: code expansion from the underlying object

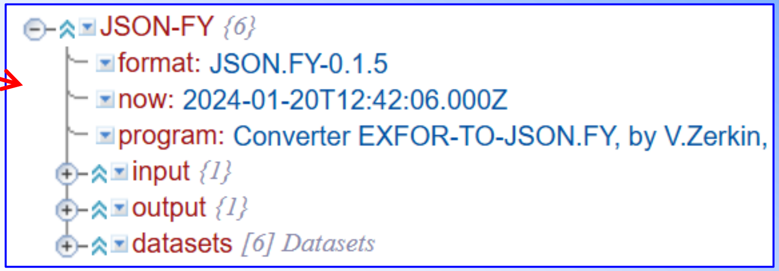
EXFOR: FreeText is shown with css:“no-wrap;mono-space” to display tabulated text and see every “space”

Fast overview the JSON structure

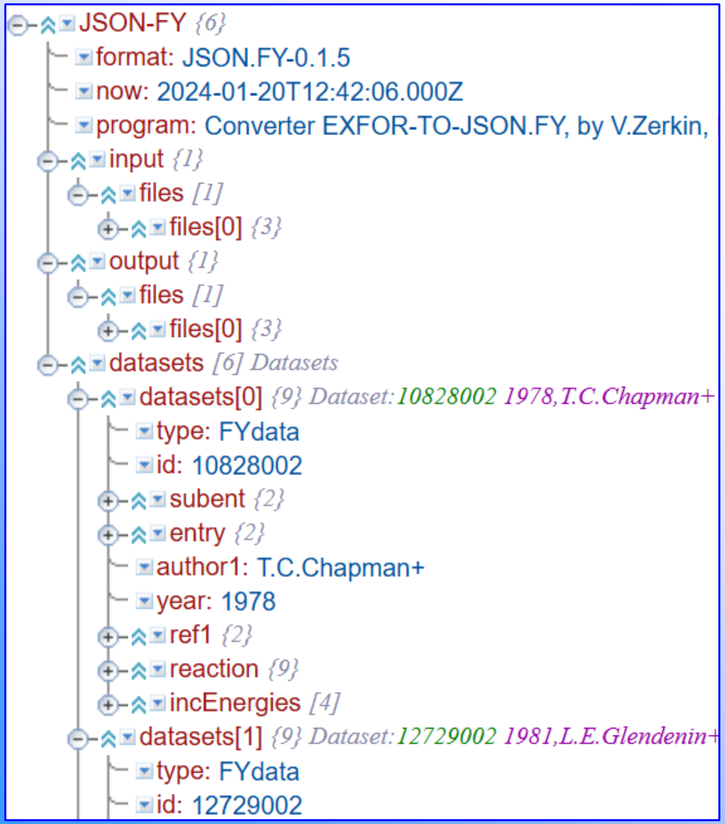
Looking on a new large JSON file, sometimes it is needed to quickly understand it's structure by opening nested levels of information. In order to do this Editor provides operations

View → open 1 level
View → open 2 levels
. . .
View → open all

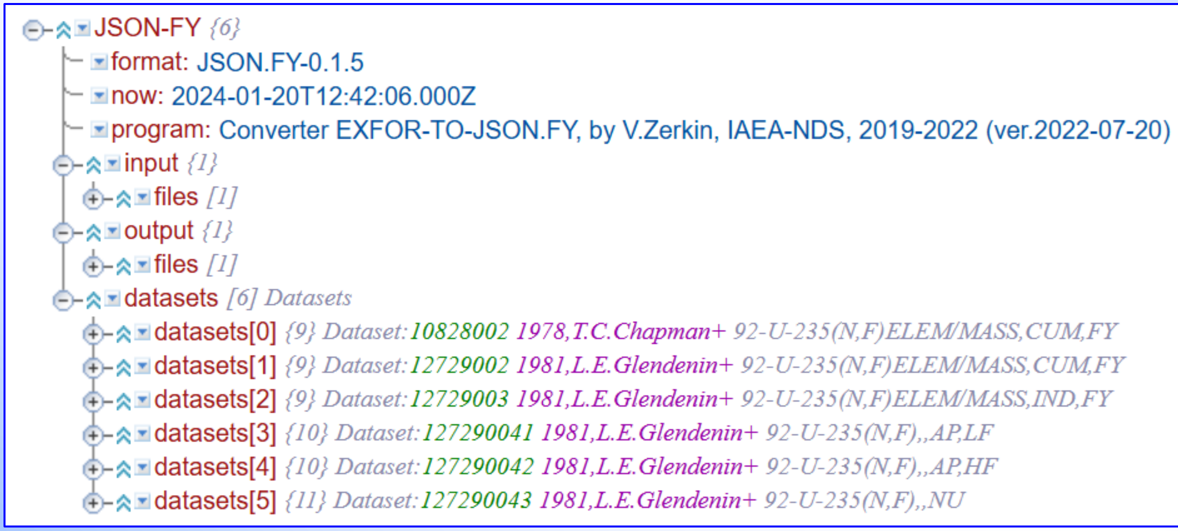




```
JSON-FY {6}
  format: JSON.FY-0.1.5
  now: 2024-01-20T12:42:06.000Z
  program: Converter EXFOR-TO-JSON.FY, by V.Zerkin,
  input {1}
  output {1}
  datasets [6] Datasets
```



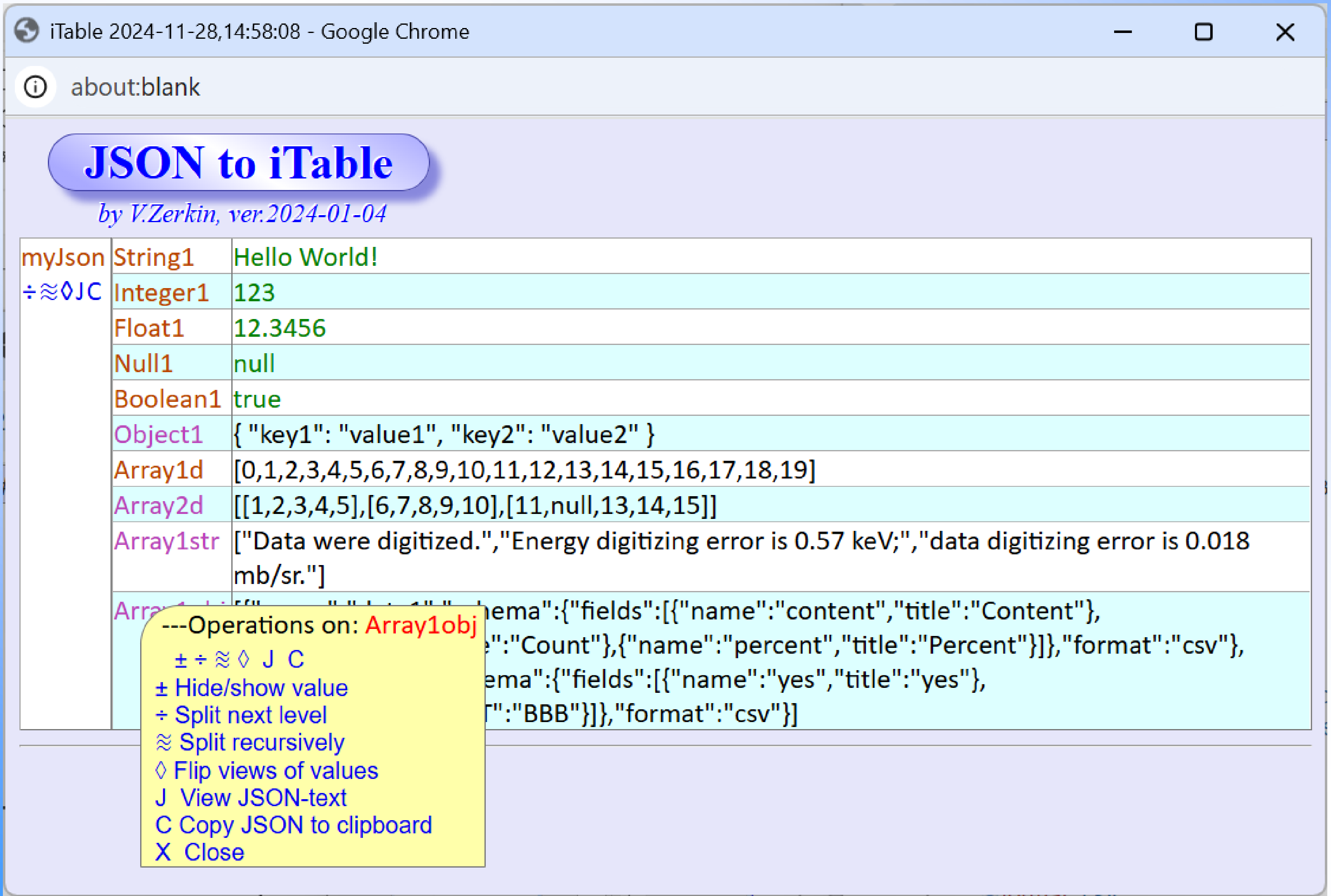
```
JSON-FY {6}
  format: JSON.FY-0.1.5
  now: 2024-01-20T12:42:06.000Z
  program: Converter EXFOR-TO-JSON.FY, by V.Zerkin,
  input {1}
    files [1]
      files[0] {3}
  output {1}
    files [1]
      files[0] {3}
  datasets [6] Datasets
    datasets[0] {9} Dataset:10828002 1978,T.C.Chapman+
      type: FYdata
      id: 10828002
      subent {2}
      entry {2}
      author1: T.C.Chapman+
      year: 1978
      ref1 {2}
      reaction {9}
      incEnergies [4]
    datasets[1] {9} Dataset:12729002 1981,L.E.Glendenin+
      type: FYdata
      id: 12729002
```



```
JSON-FY {6}
  format: JSON.FY-0.1.5
  now: 2024-01-20T12:42:06.000Z
  program: Converter EXFOR-TO-JSON.FY, by V.Zerkin, IAEA-NDS, 2019-2022 (ver.2022-07-20)
  input {1}
    files [1]
  output {1}
    files [1]
  datasets [6] Datasets
    datasets[0] {9} Dataset:10828002 1978,T.C.Chapman+ 92-U-235(N,F)ELEM/MASS,CUM,FY
    datasets[1] {9} Dataset:12729002 1981,L.E.Glendenin+ 92-U-235(N,F)ELEM/MASS,CUM,FY
    datasets[2] {9} Dataset:12729003 1981,L.E.Glendenin+ 92-U-235(N,F)ELEM/MASS,IND,FY
    datasets[3] {10} Dataset:127290041 1981,L.E.Glendenin+ 92-U-235(N,F),,AP,LF
    datasets[4] {10} Dataset:127290042 1981,L.E.Glendenin+ 92-U-235(N,F),,AP,HF
    datasets[5] {11} Dataset:127290043 1981,L.E.Glendenin+ 92-U-235(N,F),,NU
```

Additional viewer: iTable

Display JSON file as interactive table with possibility to show/hide data and option to show original text or sub-table structured by JSON key:value. Initially data displayed it table with two columns with Key and Value; if Value is complex (list or object) it will be presented as a sub-table with multiple rows.



myJson ÷≋◊JC	String1	Hello World!
	Integer1	123
	Float1	12.3456
	Null1	null
	Boolean1	true
	Object1	{ "key1": "value1", "key2": "value2" }
	Array1d	[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
	Array2d	[[1,2,3,4,5],[6,7,8,9,10],[11,null,13,14,15]]
	Array1str	["Data were digitized.","Energy digitizing error is 0.57 keV;","data digitizing error is 0.018 mb/sr."]
	Arr	[{" hema":{"fields":[{"name":"content","title":"Content"}, e":"Count"},{"name":"percent","title":"Percent"}]},"format":"csv"}, ema":{"fields":[{"name":"yes","title":"yes"}, T":"BBB"}]},"format":"csv"}]

Part III.

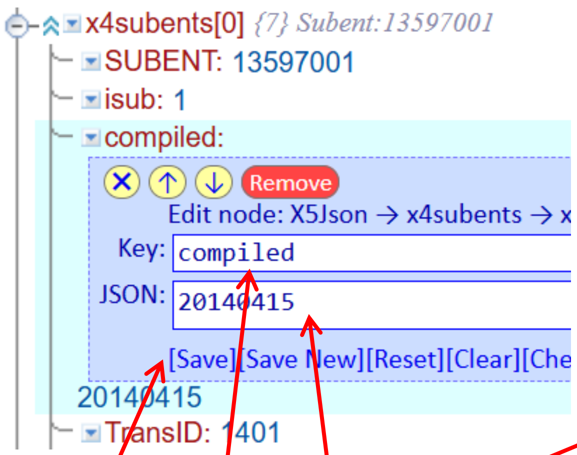
Editing. Save result.

Editing

Editing starts by Mouse-click on a key.

Basic principle. Content of every Node is JSON {key:value} – both are text and both can be edited. This means:

1) Key can be changed but must be unique on it's level (can not be the same as other keys)
2) Value: simple values (like number) and huge JSON-text can be edited in dedicated text area



x4subents[0] *{7} Subent:13597001*

SUBENT: 13597001

isub: 1

compiled:

✕ ↑ ↓ Remove

Edit node: X5Json → x4subents → x

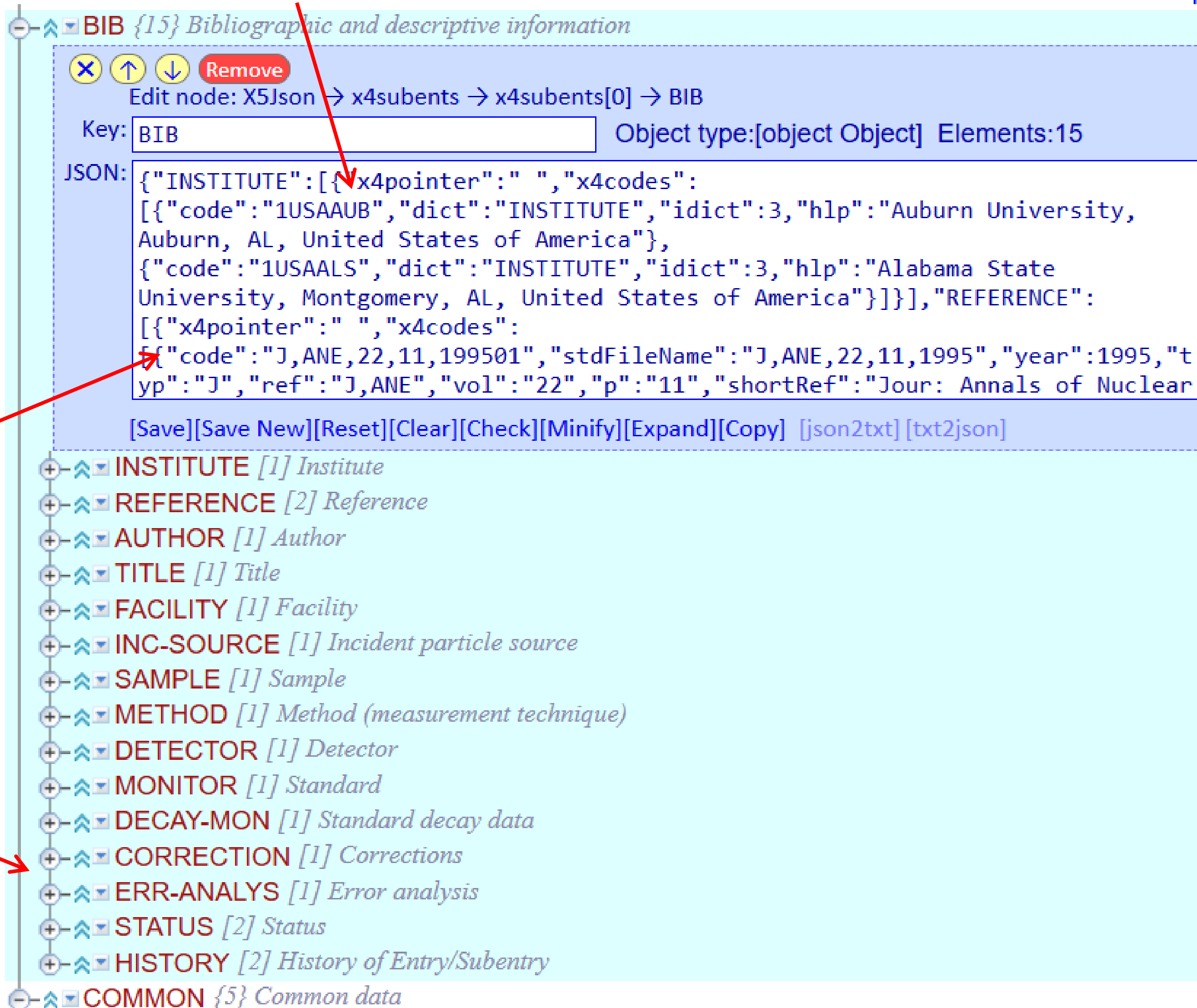
Key: compiled

JSON: 20140415

[Save][Save New][Reset][Clear][Che

20140415

TransID: 1401



BIB *{15} Bibliographic and descriptive information*

✕ ↑ ↓ Remove

Edit node: X5Json → x4subents → x4subents[0] → BIB

Key: BIB Object type:[object Object] Elements:15

JSON:

```
{"INSTITUTE":[{"x4pointer":" ","x4codes":
[{"code":"1USAAUB","dict":"INSTITUTE","idict":3,"hlp":"Auburn University,
Auburn, AL, United States of America"},
{"code":"1USAALS","dict":"INSTITUTE","idict":3,"hlp":"Alabama State
University, Montgomery, AL, United States of America"}]}],"REFERENCE":
[{"x4pointer":" ","x4codes":
[{"code":"J,ANE,22,11,199501","stdFileName":"J,ANE,22,11,1995","year":1995,"t
yp":"J","ref":"J,ANE","vol":"22","p":"11","shortRef":"Jour: Annals of Nuclear
```

[Save][Save New][Reset][Clear][Check][Minify][Expand][Copy] [json2txt] [txt2json]

INSTITUTE *[1] Institute*

REFERENCE *[2] Reference*

AUTHOR *[1] Author*

TITLE *[1] Title*

FACILITY *[1] Facility*

INC-SOURCE *[1] Incident particle source*

SAMPLE *[1] Sample*

METHOD *[1] Method (measurement technique)*

DETECTOR *[1] Detector*

MONITOR *[1] Standard*

DECAY-MON *[1] Standard decay data*

CORRECTION *[1] Corrections*

ERR-ANALYS *[1] Error analysis*

STATUS *[2] Status*

HISTORY *[2] History of Entry/Subentry*

COMMON *{5} Common data*

Edit key and value

[Save] will modify Node

[Save New] will add new Node after one which was edited.

Edited node is shown with light-cyan background (node before editing)

Move node up/down – ↑ ↓

Close node without saving - ✕

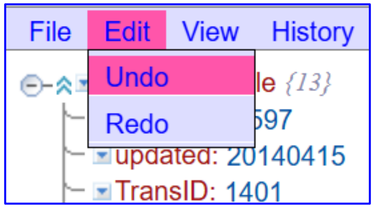
Undo/Redo. History. Restore JSON versions.

Every modification of node content or position in the current tree is recorded and can be restored by user if necessary to one or more steps. Actions can also be repeated.

1. Undo/Redo are used to cancel or repeat last operations.

   Edit → Undo

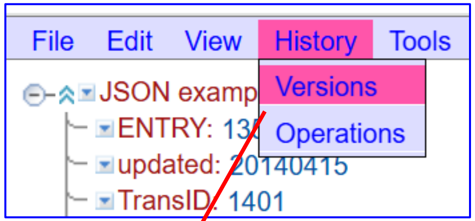
   Edit → Redo



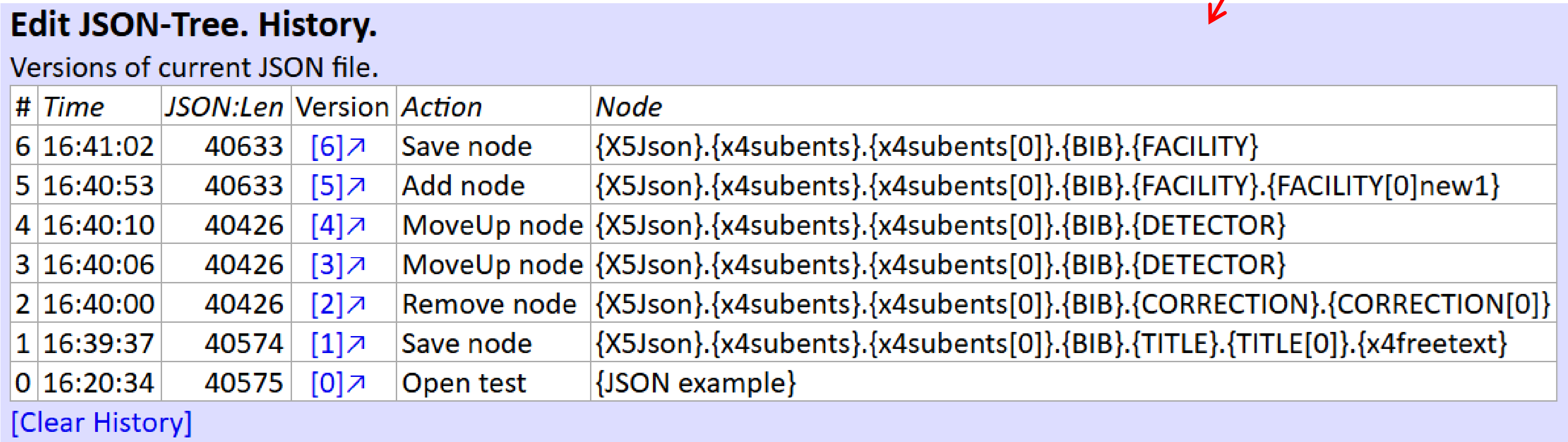
2. History shows operations which were recently done and allows to return to any step back

   History → Versions *#show what was done + return back*

   History → Operations *#show what was done*



Mouse-click on Version[number] [4]↗ will return JSON to the needed version



Edit JSON-Tree. History.

Versions of current JSON file.

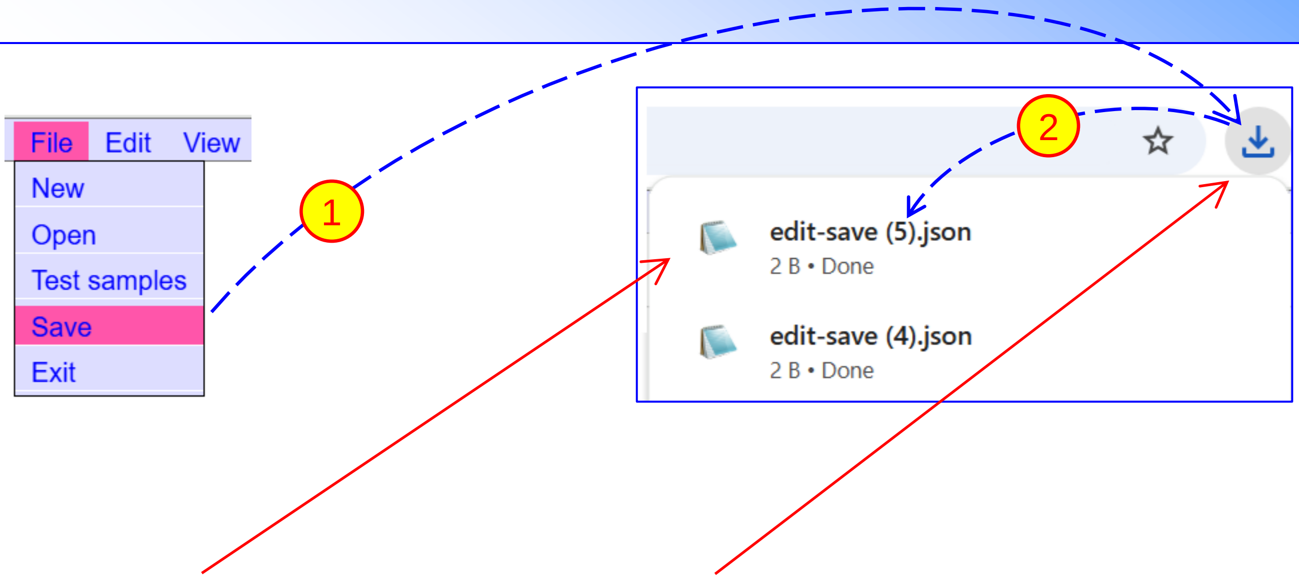
#	*Time*	*JSON:Len*	Version	*Action*	*Node*
6	16:41:02	40633	[6]↗	Save node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{FACILITY}
5	16:40:53	40633	[5]↗	Add node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{FACILITY}.{FACILITY[0]new1}
4	16:40:10	40426	[4]↗	MoveUp node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{DETECTOR}
3	16:40:06	40426	[3]↗	MoveUp node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{DETECTOR}
2	16:40:00	40426	[2]↗	Remove node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{CORRECTION}.{CORRECTION[0]}
1	16:39:37	40574	[1]↗	Save node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{TITLE}.{TITLE[0]}.{x4freetext}
0	16:20:34	40575	[0]↗	Open test	{JSON example}

[Clear History]

Save result

Output.

1) File → Save



edit-save (5).json
2 B • Done

edit-save (4).json
2 B • Done

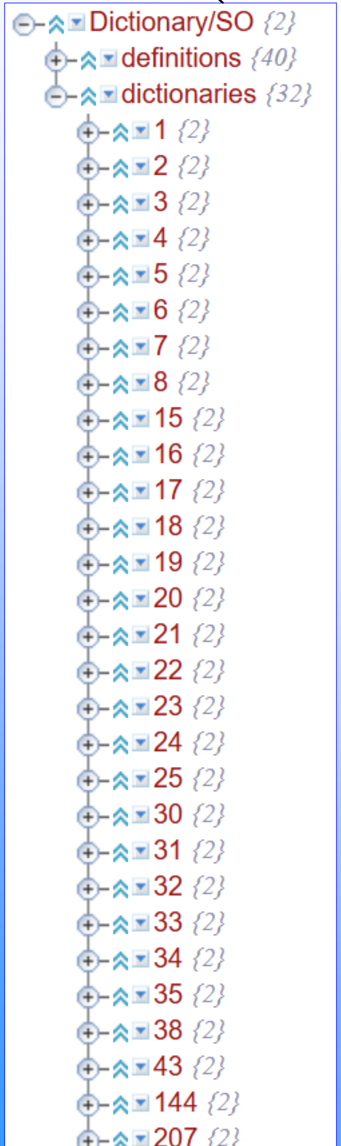
2) Now your current JSON file is stored in the Download area of your Web-Browser

Back to purpose. Viewer. Practical example.

Compare 3 Dictionary versions: https://zerkin.usite.pro/edit-json-tree/cmp3dict.htm

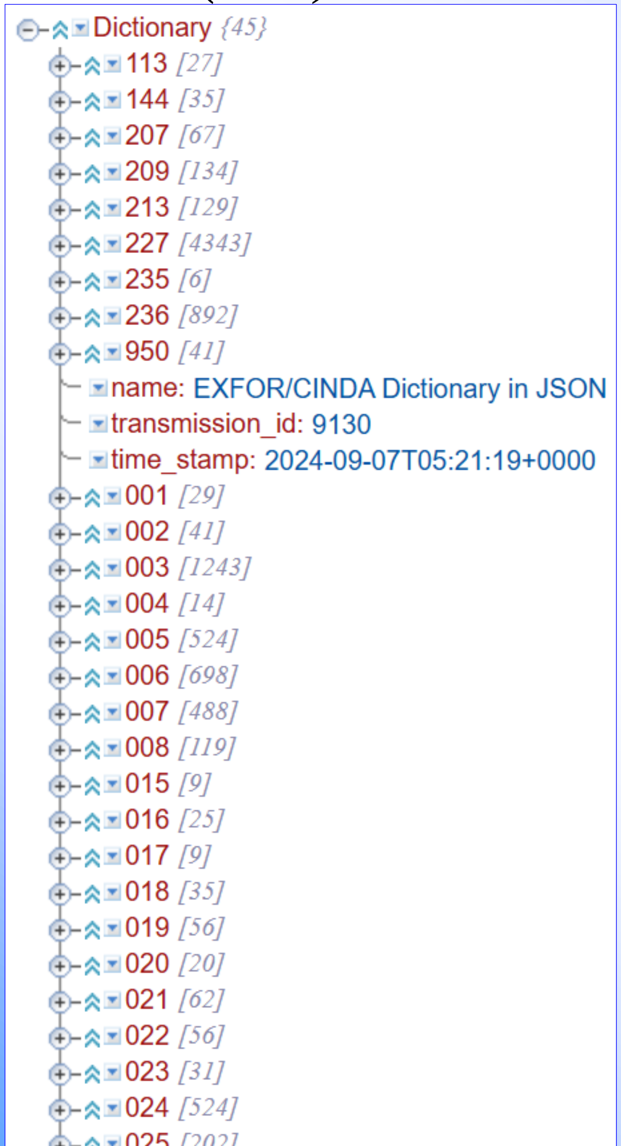
(Note. Although all three are built for different purposes and uses, we can explore technical details, see drawbacks and find bugs.)

S.Okumura(~2022)



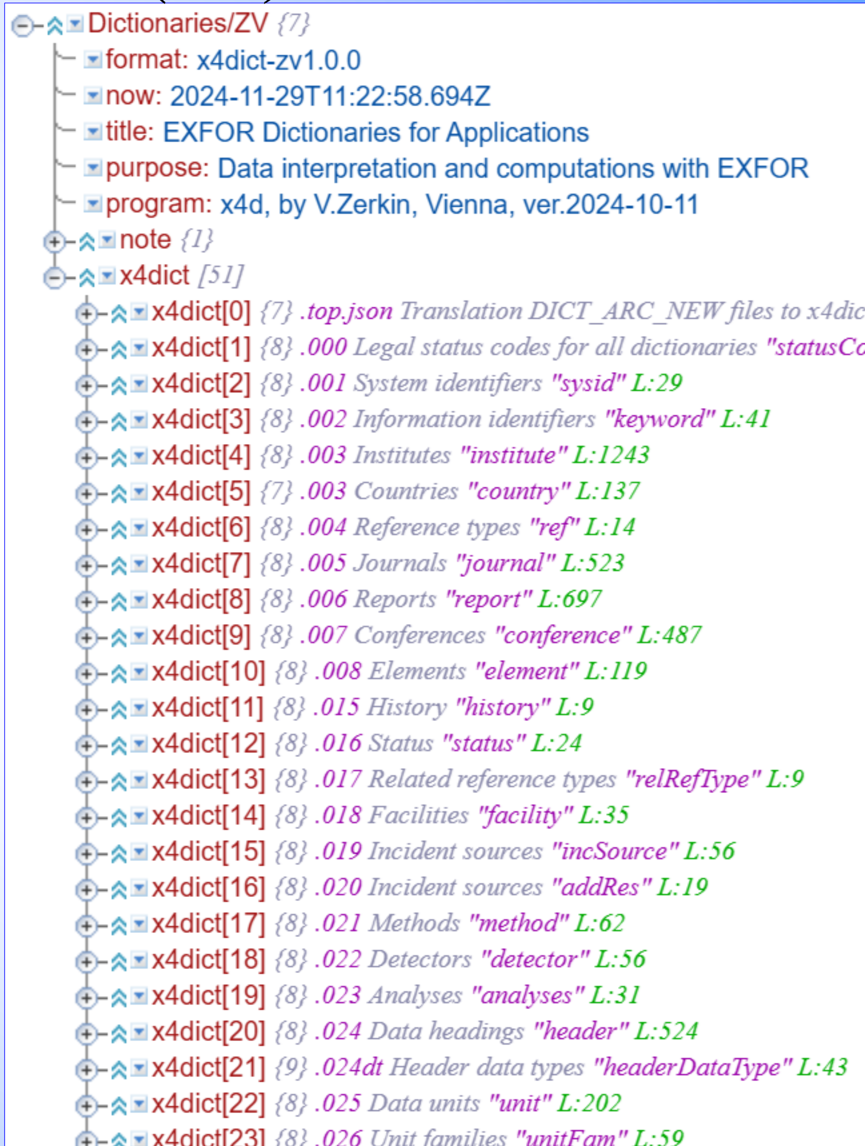
```
Dictionary/SO {2}
  definitions {40}
  dictionaries {32}
    1 {2}
    2 {2}
    3 {2}
    4 {2}
    5 {2}
    6 {2}
    7 {2}
    8 {2}
    15 {2}
    16 {2}
    17 {2}
    18 {2}
    19 {2}
    20 {2}
    21 {2}
    22 {2}
    23 {2}
    24 {2}
    25 {2}
    30 {2}
    31 {2}
    32 {2}
    33 {2}
    34 {2}
    35 {2}
    38 {2}
    43 {2}
    144 {2}
    207 {2}
```

N.Otsuka(2024)



```
Dictionary {45}
  113 [27]
  144 [35]
  207 [67]
  209 [134]
  213 [129]
  227 [4343]
  235 [6]
  236 [892]
  950 [41]
  name: EXFOR/CINDA Dictionary in JSON
  transmission_id: 9130
  time_stamp: 2024-09-07T05:21:19+0000
  001 [29]
  002 [41]
  003 [1243]
  004 [14]
  005 [524]
  006 [698]
  007 [488]
  008 [119]
  015 [9]
  016 [25]
  017 [9]
  018 [35]
  019 [56]
  020 [20]
  021 [62]
  022 [56]
  023 [31]
  024 [524]
  025 [202]
```

V.Zerkin(2024)



```
Dictionaries/ZV {7}
  format: x4dict-zv1.0.0
  now: 2024-11-29T11:22:58.694Z
  title: EXFOR Dictionaries for Applications
  purpose: Data interpretation and computations with EXFOR
  program: x4d, by V.Zerkin, Vienna, ver.2024-10-11
  note {1}
  x4dict [51]
    x4dict[0] {7} .top.json Translation DICT_ARC_NEW files to x4dic
    x4dict[1] {8} .000 Legal status codes for all dictionaries "statusCo
    x4dict[2] {8} .001 System identifiers "sysid" L:29
    x4dict[3] {8} .002 Information identifiers "keyword" L:41
    x4dict[4] {8} .003 Institutes "institute" L:1243
    x4dict[5] {7} .003 Countries "country" L:137
    x4dict[6] {8} .004 Reference types "ref" L:14
    x4dict[7] {8} .005 Journals "journal" L:523
    x4dict[8] {8} .006 Reports "report" L:697
    x4dict[9] {8} .007 Conferences "conference" L:487
    x4dict[10] {8} .008 Elements "element" L:119
    x4dict[11] {8} .015 History "history" L:9
    x4dict[12] {8} .016 Status "status" L:24
    x4dict[13] {8} .017 Related reference types "relRefType" L:9
    x4dict[14] {8} .018 Facilities "facility" L:35
    x4dict[15] {8} .019 Incident sources "incSource" L:56
    x4dict[16] {8} .020 Incident sources "addRes" L:19
    x4dict[17] {8} .021 Methods "method" L:62
    x4dict[18] {8} .022 Detectors "detector" L:56
    x4dict[19] {8} .023 Analyses "analyses" L:31
    x4dict[20] {8} .024 Data headings "header" L:524
    x4dict[21] {9} .024dt Header data types "headerDataType" L:43
    x4dict[22] {8} .025 Data units "unit" L:202
    x4dict[23] {8} .026 Unit families "unitFam" L:59
```

Understanding/comparing 3 Dictionaries.

S.Okumura(2022-2024)

No description, no timestamp

#dictionaries	32
File structure	dict*
Dictionary-struct.	dict
Dict-236 #codes	885

1

N.Otsuka(2024)

EXFOR/CINDA Dictionary in JSON

#dictionaries	42
File structure	dict
Dictionary-struct.	list
Dict-236 #codes	892

2

V.Zerkin(2024)

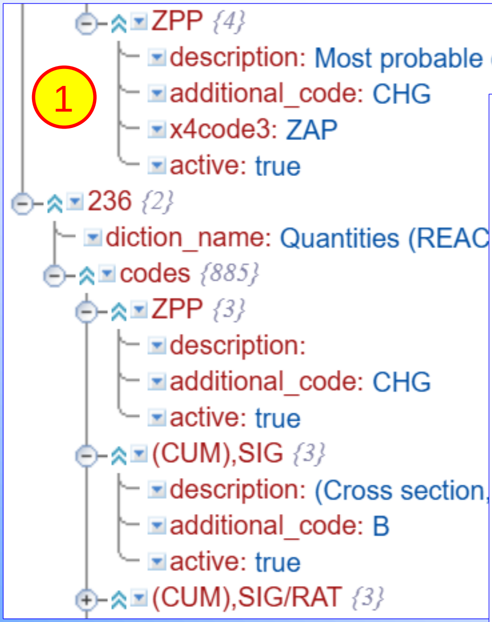
EXFOR Dictionaries for Applications

#dictionaries	51
File structure	list*
Dictionary-struct.	list
Dict-236 #codes	891

3

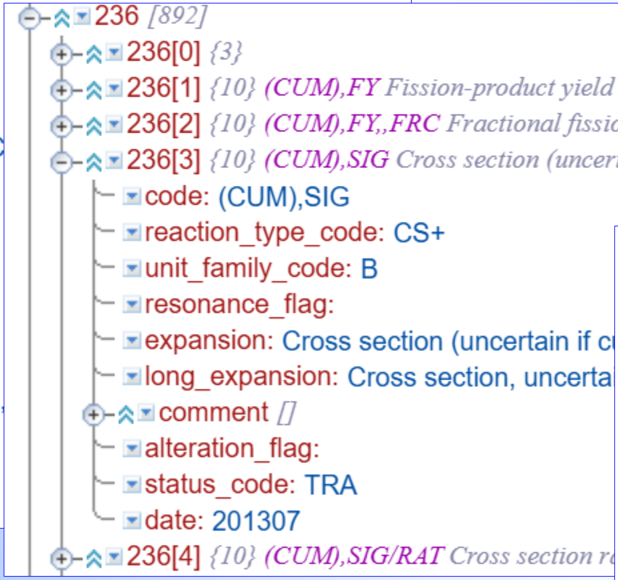
*dict - *hash table, keyed list, associative array*

*list - *ordered list of values, array, vector*



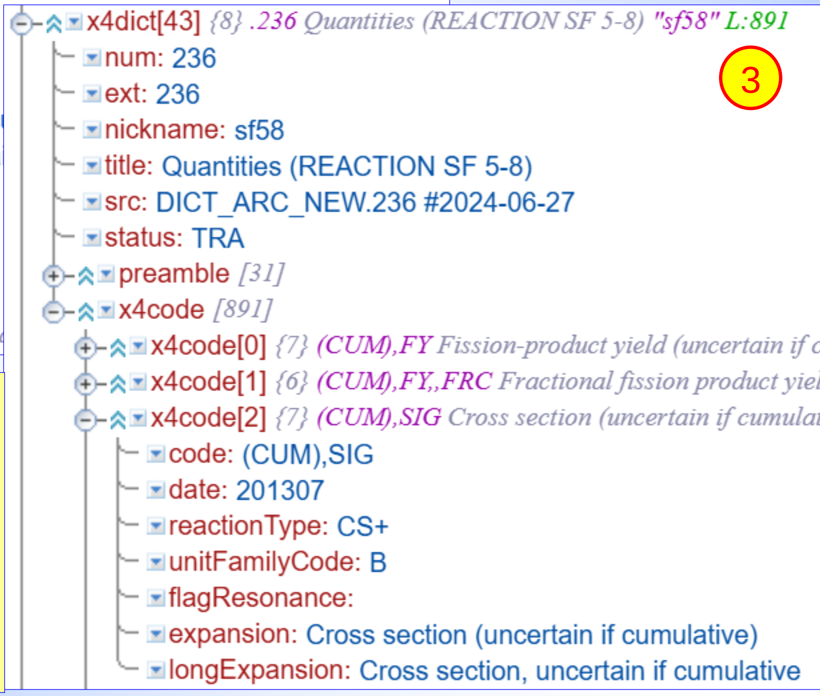
1

```
    ZPP {4}
      description: Most probable charge for given fragment energy
      additional_code: CHG
      x4code3: ZAP
      active: true
236 {2}
  diction_name: Quantities (REAC
  codes {885}
    ZPP {3}
      description:
      additional_code: CHG
      active: true
    (CUM),SIG {3}
      description: (Cross section,
      additional_code: B
      active: true
    (CUM),SIG/RAT {3}
```



2

```
236 [892]
  236[0] {3}
  236[1] {10} (CUM),FY Fission-product yield (uncertain if cumulative)
  236[2] {10} (CUM),FY,,FRC Fractional fission product yield (uncertain if cumulative)
  236[3] {10} (CUM),SIG Cross section (uncertain if cumulative)
    code: (CUM),SIG
    reaction_type_code: CS+
    unit_family_code: B
    resonance_flag:
    expansion: Cross section (uncertain if c
    long_expansion: Cross section, uncerta
    comment []
    alteration_flag:
    status_code: TRA
    date: 201307
  236[4] {10} (CUM),SIG/RAT Cross section r
```



3

```
x4dict[43] {8} .236 Quantities (REACTION SF 5-8) "sf58" L:891
  num: 236
  ext: 236
  nickname: sf58
  title: Quantities (REACTION SF 5-8)
  src: DICT_ARC_NEW.236 #2024-06-27
  status: TRA
  preamble [31]
  x4code [891]
    x4code[0] {7} (CUM),FY Fission-product yield (uncertain if c
    x4code[1] {6} (CUM),FY,,FRC Fractional fission product yiel
    x4code[2] {7} (CUM),SIG Cross section (uncertain if cumula
      code: (CUM),SIG
      date: 201307
      reactionType: CS+
      unitFamilyCode: B
      flagResonance:
      expansion: Cross section (uncertain if cumulative)
      longExpansion: Cross section, uncertain if cumulative
```

Looking to all 3, questions and comments:

C1: (1) incompleteness (e.g. Dict-236: no “*reaction type code*” given)

Q1: (1) Why ZPP appears in 235 and 236? (bug?)

Q2: (1) Why Dict-236 is shorter? (bug? GitHub: “Updated on Jul 31”)

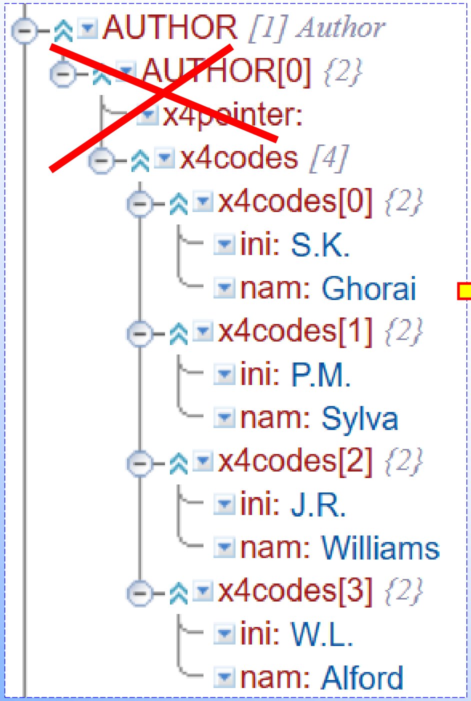
Q3: (1) D236 has 6 codes starting with “(“, but (2) and (3) – 17 (bug?)

Q4: (2) code=”” can only be [0] element of the list of codes?

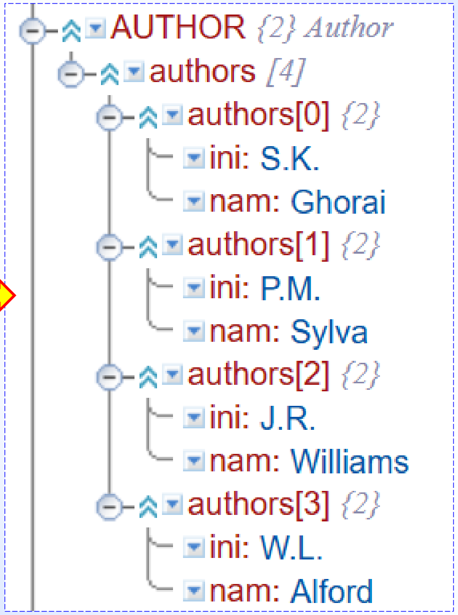
etc.

Example: proposal to change X5 structure

Edit X5json: https://zerkin.usite.pro/edit-json-tree/#1



- AUTHOR *[1] Author*
  - ~~AUTHOR[0] {2}~~
    - ~~x4pointer:~~
    - x4codes *[4]*
      - x4codes[0] *{2}*
        - ini: S.K.
        - nam: Ghorai
      - x4codes[1] *{2}*
        - ini: P.M.
        - nam: Sylva
      - x4codes[2] *{2}*
        - ini: J.R.
        - nam: Williams
      - x4codes[3] *{2}*
        - ini: W.L.
        - nam: Alford

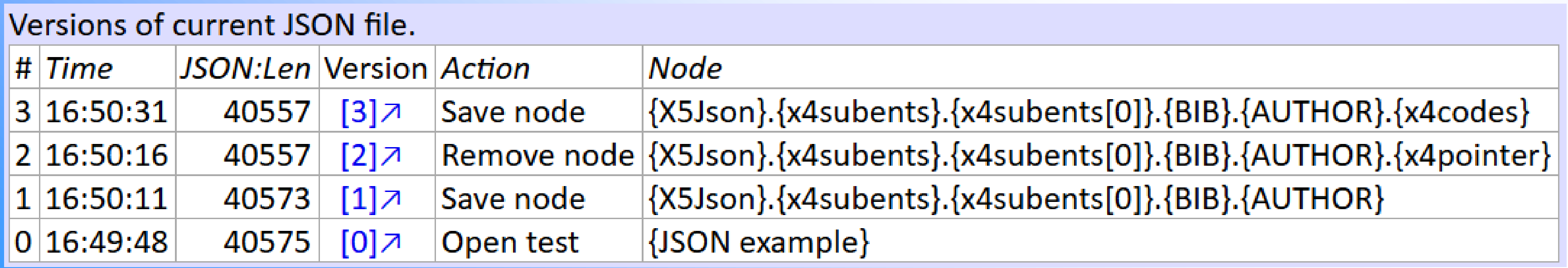


- AUTHOR *{2} Author*
  - authors *[4]*
    - authors[0] *{2}*
      - ini: S.K.
      - nam: Ghorai
    - authors[1] *{2}*
      - ini: P.M.
      - nam: Sylva
    - authors[2] *{2}*
      - ini: J.R.
      - nam: Williams
    - authors[3] *{2}*
      - ini: W.L.
      - nam: Alford

For discussion about X5json structure.
For example, we can have only one code in AUTHOR and without pointer, but X5 propose standard schema - list of codes:
KW: [x4pointer, x4codes:[{}]]
One could propose to drop one level of nesting by moving AUTHOR[0] to AUTHOR and rename “x4codes” to “authors”.

This can be done by:
1) Click on AUTHOR[0]; click on [Copy]
2) Click on AUTHOR; click [Clear], paste by <Ctrl/v>; [Save]
3) Click on x4pointer and click [Remove]
4) Click on x4codes, edit text Key: *authors*, [Save]
Note. We still need to keep “authors” under AUTHOR because we may have FreeText list.
Steps 3) ad 4) can be avoided by editing *authors* on step 1) or 2)

History of operations:

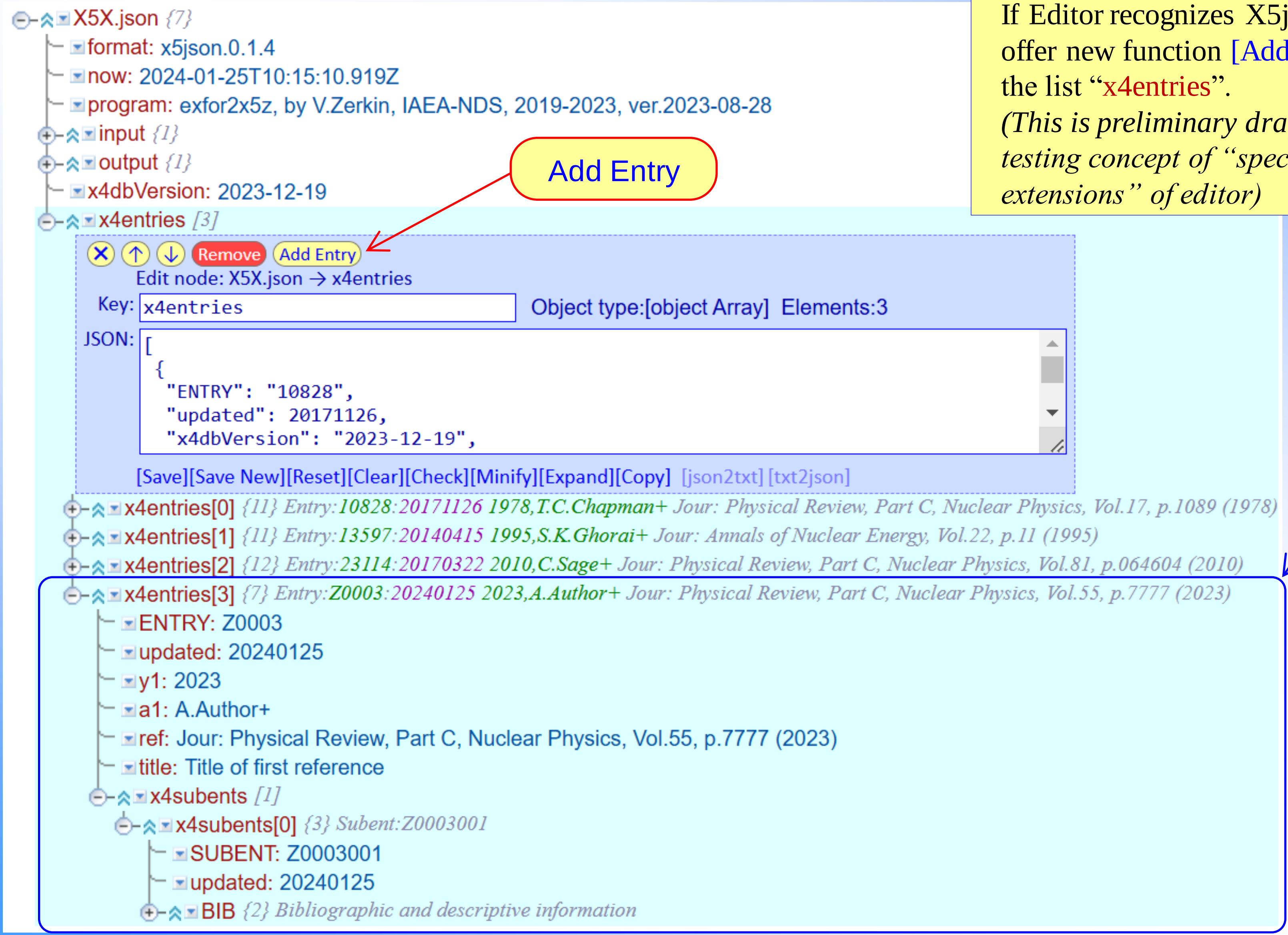


Versions of current JSON file.

#	*Time*	*JSON:Len*	Version	*Action*	*Node*
3	16:50:31	40557	[3]↗	Save node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{AUTHOR}.{x4codes}
2	16:50:16	40557	[2]↗	Remove node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{AUTHOR}.{x4pointer}
1	16:50:11	40573	[1]↗	Save node	{X5Json}.{x4subents}.{x4subents[0]}.{BIB}.{AUTHOR}
0	16:49:48	40575	[0]↗	Open test	{JSON example}

Specialized extensions of editor-part

If Editor recognizes X5json it will offer new function [Add Entry] to the list “x4entries”.
(This is preliminary draft for testing concept of “specialized extensions” of editor)



```
X5X.json {7}
  format: x5json.0.1.4
  now: 2024-01-25T10:15:10.919Z
  program: exfor2x5z, by V.Zerkin, IAEA-NDS, 2019-2023, ver.2023-08-28
  input {1}
  output {1}
  x4dbVersion: 2023-12-19
  x4entries [3]
```

Ⓧ Ⓘ Ⓘ Remove Add Entry

Edit node: X5X.json → x4entries

Key: x4entries   Object type:[object Array] Elements:3

JSON:

```
[
 {
  "ENTRY": "10828",
  "updated": 20171126,
  "x4dbVersion": "2023-12-19",
```

[Save][Save New][Reset][Clear][Check][Minify][Expand][Copy] [json2txt] [txt2json]

```
x4entries[0] {11} Entry:10828:20171126 1978,T.C.Chapman+ Jour: Physical Review, Part C, Nuclear Physics, Vol.17, p.1089 (1978)
x4entries[1] {11} Entry:13597:20140415 1995,S.K.Ghorai+ Jour: Annals of Nuclear Energy, Vol.22, p.11 (1995)
x4entries[2] {12} Entry:23114:20170322 2010,C.Sage+ Jour: Physical Review, Part C, Nuclear Physics, Vol.81, p.064604 (2010)
x4entries[3] {7} Entry:Z0003:20240125 2023,A.Author+ Jour: Physical Review, Part C, Nuclear Physics, Vol.55, p.7777 (2023)
  ENTRY: Z0003
  updated: 20240125
  y1: 2023
  a1: A.Author+
  ref: Jour: Physical Review, Part C, Nuclear Physics, Vol.55, p.7777 (2023)
  title: Title of first reference
  x4subents [1]
    x4subents[0] {3} Subent:Z0003001
      SUBENT: Z0003001
      updated: 20240125
      BIB {2} Bibliographic and descriptive information
```

New empty Entry with artificial ENTRY number

(continue editing JSON file as usual)
. . . . .

Concluding remarks

1. JSON-Tree editor is a tool to view and edit any JSON files.
   It is a full-featured editor, a universal multi-platform application running in a Web browser

2. Specialized extensions of viewer-part for nuclear data can help to understand/discuss/debug new JSON nuclear data formats

3. Specialized extensions of editor-part for nuclear data could be further explored and developed

4. The future of JSON-Tree editor is not yet determined

Thank you.