

JEventViewer 2.0

User's Guide

Carl Timmer

Jefferson Lab Experimental Physics Software
and Computing Infrastructure group

4-Apr-2021

© Thomas Jefferson National Accelerator Facility
12000 Jefferson Ave
Newport News, VA 23606
Phone 757.269.7100

Table of Contents

1.	Evio Event Viewing.....	ii
1.1	Features.....	iii
2.	File Data Viewing	5
2.1	Searching.....	6
2.1.1	By Value.....	6
2.1.2	By Location	7
2.1.3	By Page.....	7
2.1.4	By Evio Block Header	7
2.1.5	By Evio Event.....	7
2.1.6	By Evio Faults	8

Chapter 1

1. Evio Event Viewing

This manual describes a graphical user interface for looking at EVIO format files event-by-event, although it can also look at any file as a list of 32 bit integer (words). This version is compatible with evio version 6 format. To run it simply execute:

```
java org.jlab.coda.eventViewer.EventTreeFrame
```

Make sure that the EventTreeFrame class or the jar file JEventViewer-2.0.jar in is your CLASSPATH environment variable. The alternative to that is executing the provided script:

```
jeviodump
```

The following is a screen shot of the main gui after choosing a file to view.

Figure 1.1: Event-viewing gui

The screenshot shows the Jevio Event Tree application window. The top menu bar includes File, View, Dict, Event, and Filter. Below the menu is a toolbar with buttons for Event # (set to 2), Event Q, Limit (set to 3), and Size (set to 3). The event source is set to /Users/timmer/coda/evioDataFiles/compactEvioBuild.ev.lz4. The dictionary is set to from evio file.

The main area is divided into two panes. The left pane shows the EVIO event tree, which is a hierarchical structure of events. The right pane shows a hex dump of the event data, with columns for Position, +1, +2, +3, +4, +5, and Comments.

The event tree on the left shows a hierarchy starting with <Event>, which has BANKS. The BANKS node has tag=1(0x1) num=1(0x1) dataLen=5153 children=3. The BANKS node has children: BANK of INT32s, BANK of LONG64s, and BANK of TAGSEGMENTS. The BANK of INT32s node has tag=2(0x2) num=2(0x2) dataLen=1733 children=8. The BANK of LONG64s node has tag=41(0x29) num=41(0x29) dataLen=406. The BANK of TAGSEGMENTS node has tag=16(0x10) num=16(0x10) dataLen=1707. The TAGSEGMENTS node has children: TAGSEGMENT of INT32s, TAGSEGMENT of CHAR8s, TAGSEGMENT of SHORT16s, TAGSEGMENT of LONG64s, TAGSEGMENT of FLOAT32s, TAGSEGMENT of DOUBLE64s, and TAGSEGMENT of CHARSTAR8s. The TAGSEGMENT of INT32s node has tag=17(0x11) dataLen=203.

The hex dump on the right shows the data for the selected event. The first row is highlighted in yellow. The data is as follows:

Position	+1	+2	+3	+4	+5	Comments
0	0x26cec005	0x7aa9bd1e	0x7e7b9f63	0x55851d84	0x069b36cb	
5	0x2c09ec59	0x771ce26d	0x0f22e99e	0x1961701e	0x110cdfa7	
10	0x03fac74f	0x4a6333a5	0x65a59113	0x5ba6bc8c	0x21fce01c	
15	0x7dbe3c2d	0x2267d8be	0x53f11eab	0x5dc158ea	0x51dcd1dc	
20	0x3939a033	0x27dfe8ea	0x04f1f7cb	0x4889cd0f	0x76d25a86	
25	0x39765cdb	0x26191cd5	0x022e1891	0x2a2494cb	0x2b7c1614	
30	0x1ef6ec21a	0x61bea0e5	0x71f21615	0x6cb20a27	0x79557ba7	
35	0x4f35009d	0x78e5377a	0x018788a9	0x565777a8	0x58b37c4e	
40	0x601d4c0c	0x4c4f3b2e	0x28737e19	0x718a528c	0x184c10e1	
45	0x13f4254a	0x7b79f116	0x0f8c93eb	0x69469d0b	0x092ea491	
50	0x0bb9ccb3	0x6aa52aff	0x681b199f	0x8a397097	0x6b2da703	
55	0x13bbdc5c	0x1ba5c9b1	0x65069af0	0x7558b01c	0x307a4695	
60	0x0fa60a55	0x57f48f2f	0x519bb210	0x12c0e240	0x1653529f	
65	0x118312ea	0x440a10f5	0x6f1c01ef	0x787233a3	0x6b8845b3	
70	0x0cbeccadd	0x71b8d238	0x53e95875	0x2c65a120	0x15b177dd	
75	0x71da59ad	0x68d68629	0x7cbcf8c3	0x2f0b4a09	0x03ce78e3	
80	0x15239bd2	0x63257a31	0x19fc0d75	0x2962eae6	0x3e9b9286	
85	0x7b60acfe	0x228f2835	0x60da42e3	0x6b7f7d57	0x2c8f772b	
90	0x05809a5d	0x5260f0d3	0x4990b95a	0x1de70705	0x6911fe01	
95	0x3c023cd5	0x63b0fd33	0x29e5f167	0x21d87335	0x3cd4d97c	
100	0x46623b47	0x10894b77	0x03b5040c	0x1a5a135c	0x5119bd58	
105	0x250134cb	0x0562dc9c	0x2f1e7108	0x78cd18ea	0x41665ef6	
110	0x7093127f	0x49831316	0x36baf1d3	0x2e0ec78b	0x609e4332	
115	0x73c88418	0x66ab32e6	0x5422c61f	0x4fe0a361	0x764c28a3	
120	0x53839d77	0x5e920c47	0x738f0398	0x78fd3f4c	0x325d39d9	
125	0x2b88a595	0x260bcdbe	0x0351e0a5	0x4c191f22	0x2f43f79e	
130	0x1fbc066e	0x1abff443	0x15757e64	0x1aa7c1f6	0x3e5c4b35	
135	0x3f9e67b9	0x75a2a2b4	0x7f731e31	0x1ad5a454	0x08c1c09d	
140	0x7b57f126	0x32f57fe2	0x2d9373de	0x1d498a82	0x4068e8ce	
145	0x2f43b49f	0x415695e9	0x4d291ffc	0x3bda22c0	0x32cec3fa	
150	0x748c4087	0x0818b59e	0x42765210	0x5f2f8f89	0x092213ce	
155	0x730a591e	0x75e343d2	0x14ae92de	0x42b0490d	0x5c1ef15ae	
160	0x6236af1a	0x4e6d238f	0x1425b6a2	0x4d1b4b43	0x00ac24fb	
165	0x3ca0d12e	0x596ec858	0x7c5cf751	0x2118308b	0x5aac8550	
170	0x62f88706	0x5c4f706c	0x35f336dc	0x480f8389	0x0d2e94c0	
175	0x09610728	0x06298703	0x1c7d696d	0x277b2d9a	0x5677e619	
180	0x4b588a15	0x7bcc2e90	0x699c27a7	0x37e711d3	0x28bb3c24	
185	0x28292485	0x3777f2fd	0x4ce368ea	0x11c5cf45	0x52cdf6e2	
190	0x737299ae	0x4778a3f1	0x5b01658c	0x6b35ab3a	0x1f42bf0d	
195	0x727fe557e	0x3a26570d	0x2cd1a120	0x0b46a375	0x190cea2f3	
200	0x74ae8edb	0x2b3e9229	0x54591cb2			

At the bottom of the window, there are fields for version (6), structure (TAGSEGMENT), tag (17), length (812 bytes), compression (Lz4), data type (INT32), number (0), and description.

1.1 Features

Here's a quick list of the main features:

- Valid event sources are files, cMsg messages, and ET buffers
- Fast compare ability for data from different events
- When receiving events through cMsg or ET, they can be filtered based on their CODA event type (physics, control, etc.) and trigger type if physics event
- View integer data as hex or decimal
- Select dictionary from event source or from separate file containing dictionary
- View the dictionary being used
- Export any evio file in xml format
- View the contents of any file as 32 bit hex integers
- Search for values, positions, evio records/blocks, evio events, or evio errors

In the figure above, starting with the middle of the gui first, the left side shows a tree structure diagram of the whole, single evio event being viewed. Notice that the type of each evio structure is given (bank, segment, tagsegment), along with the type of data it contains, tag, num, size, and # of children. Tag and num are shown in decimal and hex. If a dictionary is being used, the dictionary name is displayed instead of the corresponding structure type, data type, tag, and num values.

The right side, on the other hand, shows the data of any selected bank, segment, or tagsegment that contains a data type and not another container type. Integers can be displayed in hex or decimal.

A fast compare feature is able to compare data from different events. If the current event is changed while viewing the data of its selected structure, and if the new event has a structure with the same hierarchy of tags that the previous selection had, it too is automatically selected. This facilitates comparing the same structure in each successive event by simply hitting the “next” event button.

A dictionary can be loaded from a separate xml format file, or it can come embedded in an evio format file or buffer (cMsg, ET). The viewer allows the user to switch, in the “Dict” menu, between the different dictionaries if more than one is available. Any dictionary being used can be displayed instead of the data.

Selecting an ET system or a cMsg server as an event source, in the “Event” menu, brings up other menus to allow the proper connections to be created and maintained. The only assumptions made are that in a cMsg message, the evio data is contained in the byteArray field. Any dictionary is first looked for in the evio data and if none is found, it is looked for in a String payload item called “dictionary”.

The box in the upper left (under the row of menu buttons), “Event #”, shows the event currently selected (in this case 2) and allows the user to navigate to the desired event.

The box to its right, “Event Q”, shows different things depending on if the data source is a file, cMsg message, or ET event. For files, it shows the total number of events (in this case 3). For cMsg messages and ET events, on the other hand, events are continually arriving. In this case, “Size” shows the number of events currently in an internal queue. “Limit” allows the user to set the size of this internal queue, while “Clear” will remove all events currently in the queue. Once this queue is full, nothing else is added. The “Event #” controls can be used to switch between events in the queue.

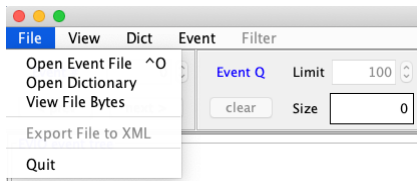
Switching between the different event sources can be done in the “Event” menu item. When selecting a cMsg or ET source, the “Filter” menu is enabled. With this menu, the user can choose to look at control, partially-built physics, physics events, or any combination as well as the selecting the run type of interest.

Notice that above the data, there are boxes containing the event and dictionary sources. Beneath the data are boxes containing information about the selected data structure such as its structure type, data type, tag, num, length in bytes, description, evio version, and the type of data compression if any.

Warning about performance: for large files, make sure they are local to the machine that’s running this program since it uses memory mapping to look at file data. You do not want the performance hit you’ll take for viewing files which are served over the network!

Chapter 2

2. File Data Viewing



The following figure is a screen shot of a file's data obtained by selecting the “View File Bytes” option of the “File” menu of the initial screen shown previously.

Figure 2.1: Data-viewing gui

The screenshot shows the 'compactEvoBuild.ev bytes' window. The 'File Info' panel on the left displays details for 'compactEvoBuild.ev bytes', including File ID (0x4556494f), File type (EVO_FILE), File split (1), Header words (56), Record count (2), Index array bytes (0), Evio version (6), Has dictionary (true), Has first event (false), Has trailer & index (true), User header bytes (620), User register (0x0), Trailer position (62660), Evio int 1 (0x0), and Evio int 2 (0x0). The 'Search By' section has radio buttons for Word Value, Word Position, Page Scrolling, Evio Record (selected), Evio Event, and Evio Fault. The 'Search For' field contains '0xc0da0100'. The 'Search Controls' section has 'Start Scan' and 'Stop' buttons. The 'Record Info' panel shows Total words (10326), Record # (1), Header words (14), Event count (2), Index array bytes (8), Version (6), Has dictionary (false), Is last (false), User header bytes (0), Uncompressed bytes (41240), Compression type (None), Compressed words (0), User register 1 (0x0), and User register 2 (0x0). The main table displays word positions and values, with a 'Color Key' on the right indicating different data types: File (Header, Index array, User header), Record Normal (Header, Index array, User header), Errors (Record with error, Event with error, Evio struct error), Event normal, Word value, and Current selection. The table shows a sequence of words and their values, with some words highlighted in yellow to indicate errors or specific data types.

There are occasions when one wants to examine the raw bytes in a file. This tool will allow one to do just that. It is capable of viewing any file's data, although it's designed specifically to look at evio version 4 and 6 format data.

Each cell of the table contains 32 bits worth of data displayed in hex. Data can be switched between big and little endian under the "File" menu. The table contains up to 1GB worth of data at one time. For larger files, the next or previous 1GB are loaded when required while scanning through it. On the immediate right of the data is a slider which indicates where the current view is in relation to the part of the file that is currently memory mapped (up to 1GB). On the far right is a color key to highlighted cells.

The figure above is showing an evio version 6 format file. All such files have a file header shown in blue. The light blue is the main header of 14 words. Although there is no index in this case, there is a dictionary which is stored in the file header's so-called user header. This is seen highlighted in dark blue. In the "File Info" box on the top left, all values in the file header appear in a table.

When searching for record headers, each one shows up highlighted in green. The light green is the main header of 14 words. The mandatory index of events shows up in medium green. Although not seen above, since it isn't used in evio, any associated user header is shown in dark green. When a record header is found, it's data is shown in the "Record Info" box on the left.

When searching for events, the first 2 words of each are highlighted in cyan. When an event is found, it's data is shown in the "Event Info" box on the left (not seen in the figure above).

2.1 Searching

In order to facilitate finding the data of interest, there are a number of different ways to hunt through it. The control panel on the left has "Search By" radio buttons allowing one to select whether to search by:

1. Looking for a given value
2. Jumping to a given position in the file
3. Scrolling page by page or by blocks of 40 pages
4. Jumping from one evio record/block header to the next
5. Jumping from one evio event to the next
6. Scanning the whole file for evio faults or errors

2.1.1 By Value

Look for a given value by selecting the "Word Value" radio button, typing the value into the "Search For" widget, and then hit the forward or backward search button under "Search Controls". The "Stop" button will be activated since searching a large file (say 20GB) may take extended time. If a search is stopped, the view position stays where it

was when the search was started. If stopped, starting another search starts from the same location. A progress bar is there to estimate how much of the file has been searched.

When a value is found, it is highlighted in gold. Hit the search button again to find the next or previous value. Highlights can be cleared under the “File” menu.

2.1.2 By Location

Look at a given location in the file by selecting the “Word Position” button, typing the position into the “Search For” widget, and then hitting the “Go” button. The view jumps to the given location and the value is selected (but not highlighted). The first position starts at 1, not 0. You can read the position from the table by taking the number in the far left column and adding the number of the heading at the very top of the column.

2.1.3 By Page

The “Page Scrolling” button activates the “<” and “>” buttons which hop through the file page (or view) by page. It also activates the “<<” and “>>” buttons immediately underneath which move through the file in 40 pages at a click.

2.1.4 By Evio Record/Block Header

For evio version 4 files: look for an evio format block header by selecting the “Evio Block” button. The program first looks for the magic # (0xc0da0100) of an evio block header. If found, it checks that the header length is 8 words. If so, it highlights all 8 words in green. All the information contained in that header is also displayed on the left in a panel called “Block Info.”

For evio version 6 files: look for an evio format block header by selecting the “Evio Record” button. The program first looks for the magic # (0xc0da0100) of an evio record header. If found, it checks that the header length is 14 words. If so, it highlights all 14 words in light green. It highlights the index part of the header in medium green, and the user header part in the darkest green. All the information contained in that header is also displayed on the left in a panel called “Record Info” which can be seen in the figure above.

2.1.5 By Evio Event

Look for an evio event (top level evio bank) by selecting the “Evio Event” button. This is less straightforward than looking for record/block headers since there is no universal signature to look for. There are two ways to do the search. The first way is start the search immediately upon loading the file’s data or to first select a position before any events. Then hit the forward button. It is smart enough to hop over any file/record/block headers encountered and uses the length found in the event’s header to be able to find the next one when the forward button is clicked again. The first two words (or header) of each event found in this way is highlighted in cyan and the header information is displayed on the left in a panel called “Event Info” (see figure below).

Event Info	
Length	4
Tag	0xffd1
Num	0
Type	BANK
Data type	UINT32
Padding	0
Bank Type	Prestart event

2.2 Event information panel

The second way to search is to select the known first word of an event with the mouse. Hit the forward button to find subsequent events. Remember that the word immediately after a record/block header is the first word of an event. Hint: selecting the first word of any bank structure (top level or not) will display all of its information

A quick note on the bank type. In CODA online, some tags are reserved for specific purposes. If a selected event has such a reserved tag, its purpose will be shown as the “Bank Type”.

2.1.6 By Evio Faults

Look for faults or errors in the evio format by selecting the “Evio Fault” button. Simply hit the “Start Scan” button and this program scans the file from beginning to end (or as far as it can parse) and lists all blocks containing errors in a panel on the left called “Evio Errors” (which can be seen in figure 2.3 below).

The algorithm used to find these errors tries to parse as much of the file as possible. For example, if a block header length does not equal the sum of the lengths of all the events it contains, then the block header length is assumed for the moment to be correct and the event lengths in error. It tries to continue by scanning the next block and stops if it encounters an unrecoverable error or makes it to the end of the file.

Errors that are caught include bad/inconsistent values in a block/event header, wrong endianness of the displayed data, length of block header not consistent with length of contained events, and not enough data to read block/event (usually a bad length), and too large of an event count in a header. The search can go into events themselves to find lower level evio errors.

For an evio version 6 file, it will find inconsistencies between compression type and header values of compression word length and uncompressed data length. Any conflict between the index length and the number of events in a record will be flagged. Of course, if a file contains compressed data, evio events will not be scanned.

To print out suspicious record numbers or record header sizes, one must set the debug flag by hand in the scanFileForErrors() method of the EvioScanner(V6).java file.

Each block in which there is a problem is listed as a button. Click one and it hops to the beginning of that block which will be highlighted in red. Within that block, the “>” and “<” buttons move from event to event. If an event has an error, it is the last event to be

accessible through the search buttons and will be highlighted in purple. If the event containing the error has an internal bank or structure with an error, it can also be accessed through the search buttons and will be highlighted in orange. A corresponding error message (or messages) is displayed at the top of the gui in red text.

Below, a small file with evio format errors has been scanned. It reveals errors in 2 records. The first record is selected showing, in red, a header with an uncompressed data length of 0 even though there is no compression. It also shows the header saying it contains 3 events but there are entries in the index for only 2. Finally, it found an error in the first event, signified by its header in purple. The error is in a sub-structure, highlighted in orange. In this case a little investigation shows that the second bank header word shows padding of 2 for a data type of 32 bit unsigned int, when it should be 0.

Figure 2.3: Error Scanning

The screenshot displays the 'HandCreatedV6.ev bytes' application window. The main area shows a list of words with their positions and hexadecimal values. The first word, at position 0, is highlighted in red and labeled 'File Header'. The second word, at position 5, is highlighted in purple and labeled 'File Header'. The third word, at position 10, is highlighted in orange and labeled 'File Header'. The fourth word, at position 15, is highlighted in orange and labeled 'File Header'. The fifth word, at position 20, is highlighted in orange and labeled 'File Header'. The sixth word, at position 25, is highlighted in orange and labeled 'File Header'. The seventh word, at position 30, is highlighted in orange and labeled 'File Header'. The eighth word, at position 35, is highlighted in orange and labeled 'File Header'. The ninth word, at position 40, is highlighted in orange and labeled 'File Header'. The tenth word, at position 45, is highlighted in orange and labeled 'File Header'. The eleventh word, at position 50, is highlighted in orange and labeled 'File Header'. The twelfth word, at position 55, is highlighted in orange and labeled 'File Header'. The thirteenth word, at position 60, is highlighted in orange and labeled 'File Header'. The fourteenth word, at position 65, is highlighted in orange and labeled 'File Header'. The fifteenth word, at position 70, is highlighted in orange and labeled 'File Header'. The sixteenth word, at position 75, is highlighted in orange and labeled 'File Header'. The seventeenth word, at position 80, is highlighted in orange and labeled 'File Header'. The eighteenth word, at position 85, is highlighted in orange and labeled 'File Header'. The nineteenth word, at position 90, is highlighted in orange and labeled 'File Header'. The twentieth word, at position 95, is highlighted in orange and labeled 'File Header'. The twenty-first word, at position 100, is highlighted in orange and labeled 'File Header'.

File Info

- File ID: 0x4556494f
- File type: EVIO_FILE
- File split #: 1
- Header words: 56
- Record count: 3
- Index array bytes: 16
- Evio version: 6
- Has dictionary: false
- Has first event: false
- Has trailer & index: true
- User header bytes: 12
- User register: 0x807060504030201
- Trailer position: 348
- User int 1: 0x0
- User int 2: 0x0

Search By

- ☐ Word Value
- ☐ Word Position
- ☐ Page Scrolling
- ☐ Evio Record
- ☐ Evio Event
- ☒ Evio Fault

Search For

0xc0da0100

Evio Errors

- ☒ Block 1
- ☐ Block 2

Search Controls

< >

Start Scan Stop

Done

Record Info

- Total words: 31
- Record #: 1
- Header words: 14
- Event count: 3
- Index array bytes: 8
- Version: 6
- Has dictionary: false
- Is last: false
- User header bytes: 12
- Uncompressed bytes: 0
- Compression type: None
- Compressed words: 0
- User register 1: 0x0
- User register 2: 0x0

Color Key

- File**
 - Header
 - Index array
 - User header
- Record Normal**
 - Header
 - Index array
 - User header
- Errors**
 - Record with error
 - Event with error
 - Evio struct error
- Event normal
- Word value
- Current selection