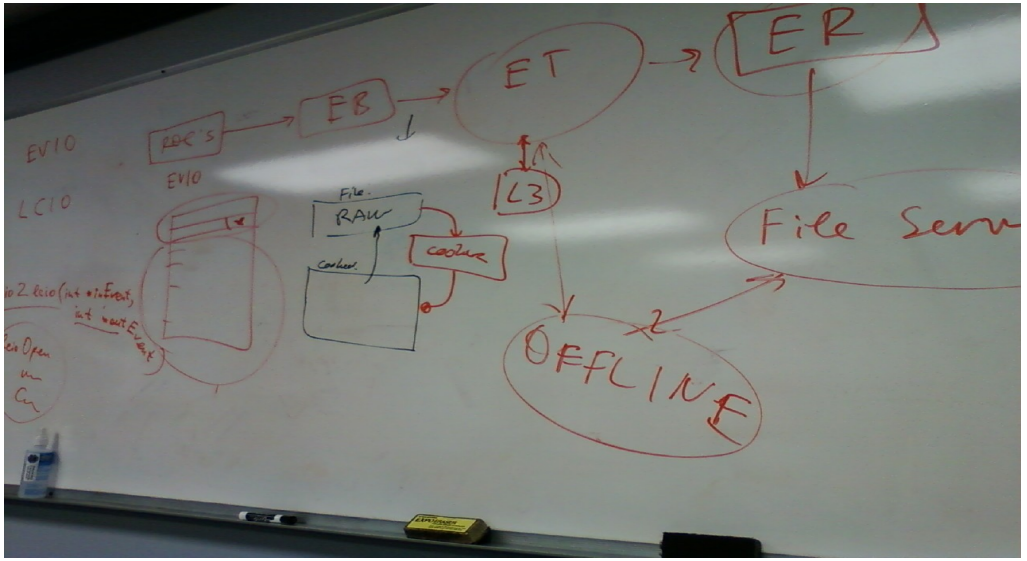
Status of EVIO / LCIO integration

For the 17 Oct 2011
HPS Software Mtg.
@ JLAB

By
Homer NEAL (SLAC)

EVIO/LCIO Interface

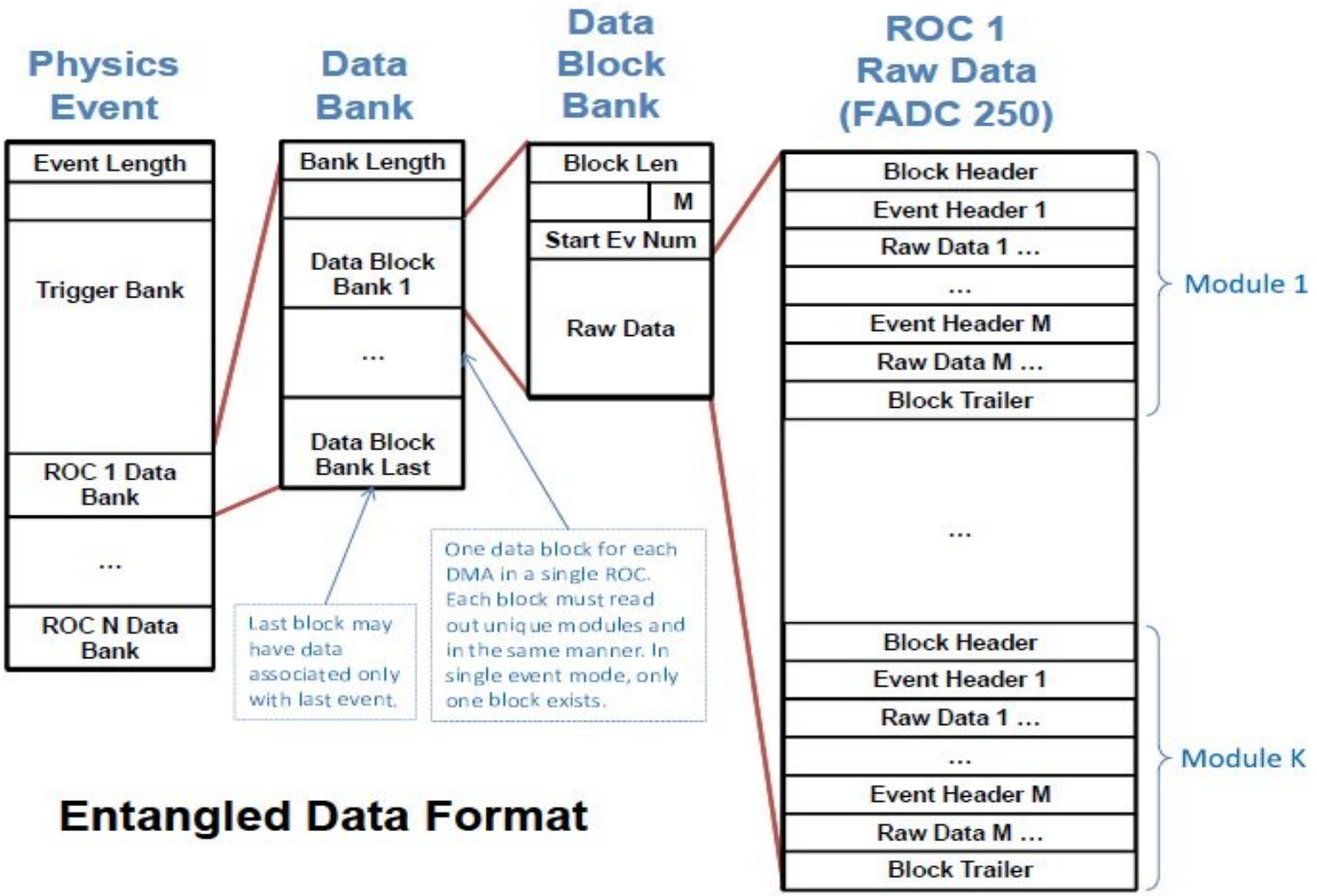
- Raw data will be presented in EVIO format
  - EVIO is the Hall B online existing and trusted format
- Reconstruction will use LCIO format
  - Format used by LCSIM
  - Random access



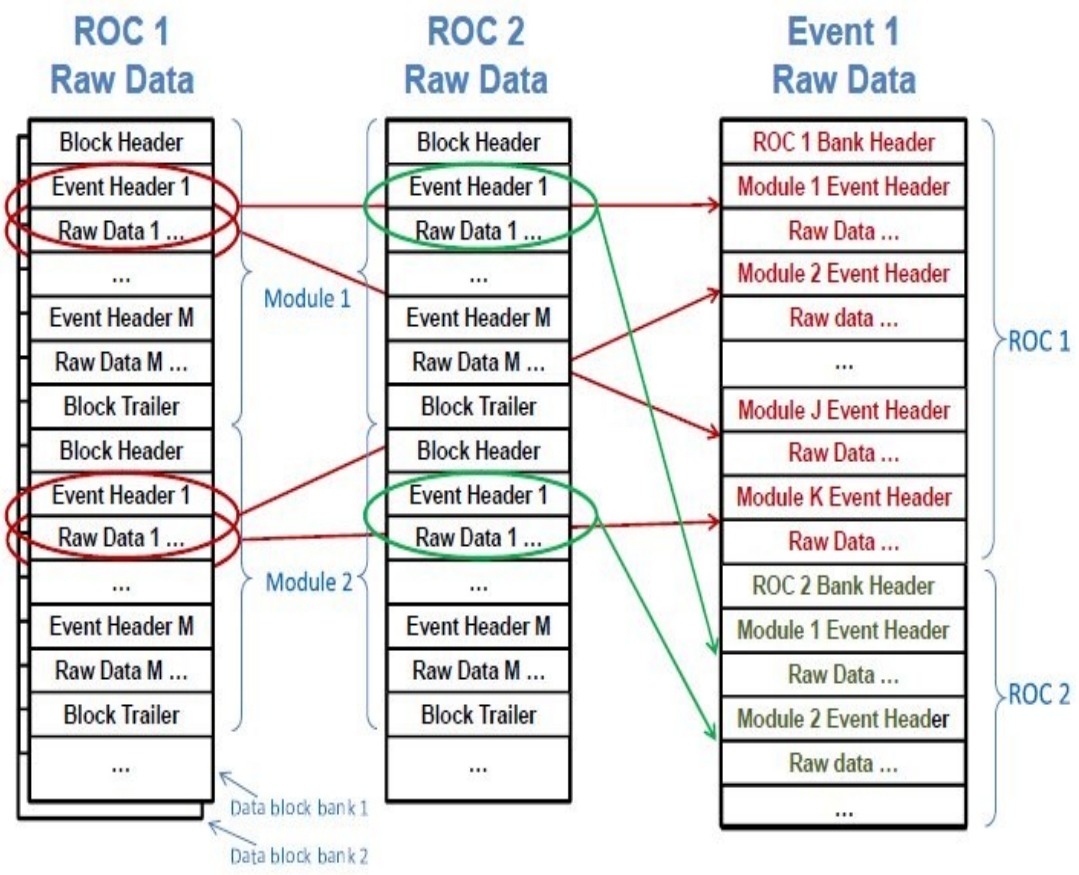
Conversion – Need to know

- Are the data from all subsystems from a given trigger presented as a single event?
- How to access specific data elements from the EVIO blocks?
- What is contained in the headers?
- metadata

Entangled Data Format



Disentangled Data Format



Entangled To Disentangled FADC 250 Raw Data

Window Raw Data (4) – raw ADC data samples for the trigger window. The first word identifies the channel number and window width. Multiple continuation words contain two samples each. The earlier sample is stored in the most significant half of the continuation word. Strict time ordering of the samples is maintained in the order of the continuation words. A sample not valid flag may be set for any sample; for example, the last reported sample is tagged as not valid when the window consists of an odd number of samples.

Word 1:
(31) = 1
(30 – 27) = 4
(26 – 23) = channel number (0 – 15)
(22 – 12) = reserved (read as 0)
(11 – 0) = window width (in number of samples)

Words 2 - N:
(31) = 0
(30) = reserved (read as 0)
(29) = sample x not valid
(28 – 16) = ADC sample x (includes overflow bit)
(15 – 14) = reserved (read as 0)
(13) = sample x + 1 not valid
(12 – 0) = ADC sample x + 1 (includes overflow bit)

Window Sum (5) – sum of the raw data samples for the trigger window. Pedestal subtraction may be included.
(31) = 1
(30 – 27) = 5
(26 – 23) = channel number (0 – 15)
(22) = window sum overflow flag
(21 – 0) = window raw data sum

An Event

FADC 250

Data Type Values	
0 – block header	7 – pulse integral
1 – block trailer	8 – pulse time
2 – event header	9 – streaming raw data
3 – trigger time	10 – 12 user defined
4 – window raw data	13 – event trailer (debug only)
5 – window sum	14 – data not valid (empty module)
6 – pulse raw data	15 – filler (non-data) word

Block Header Word Format

Bits	Value	Comment
31	1	This is a type defining word
30 – 27	0	Data type = block header
26 – 22	Slot ID	Set by VME64 backplane
21 – 14	Event #	Number of events in block
13 – 12	Module Type	0=FADC250, etc.
11 – 0	Event block #	Used to align block when building events

General Data Word Format

31st bit	Bits	Usage
1	30 - 27	4-bit data type (see chart)
1	26 - 0	Data type dependent data payload
0	30 - 0	Data payload using last defined data type

Block Trailer Word Format

Bits	Value	Comment
31	1	This is a type defining word
30 – 27	1	Data type = block trailer
26 – 22	Slot ID	Set by VME64 backplane
21 – 0	Total # of words in block of events	Number of 32 bit words in block

Event Header Word Format

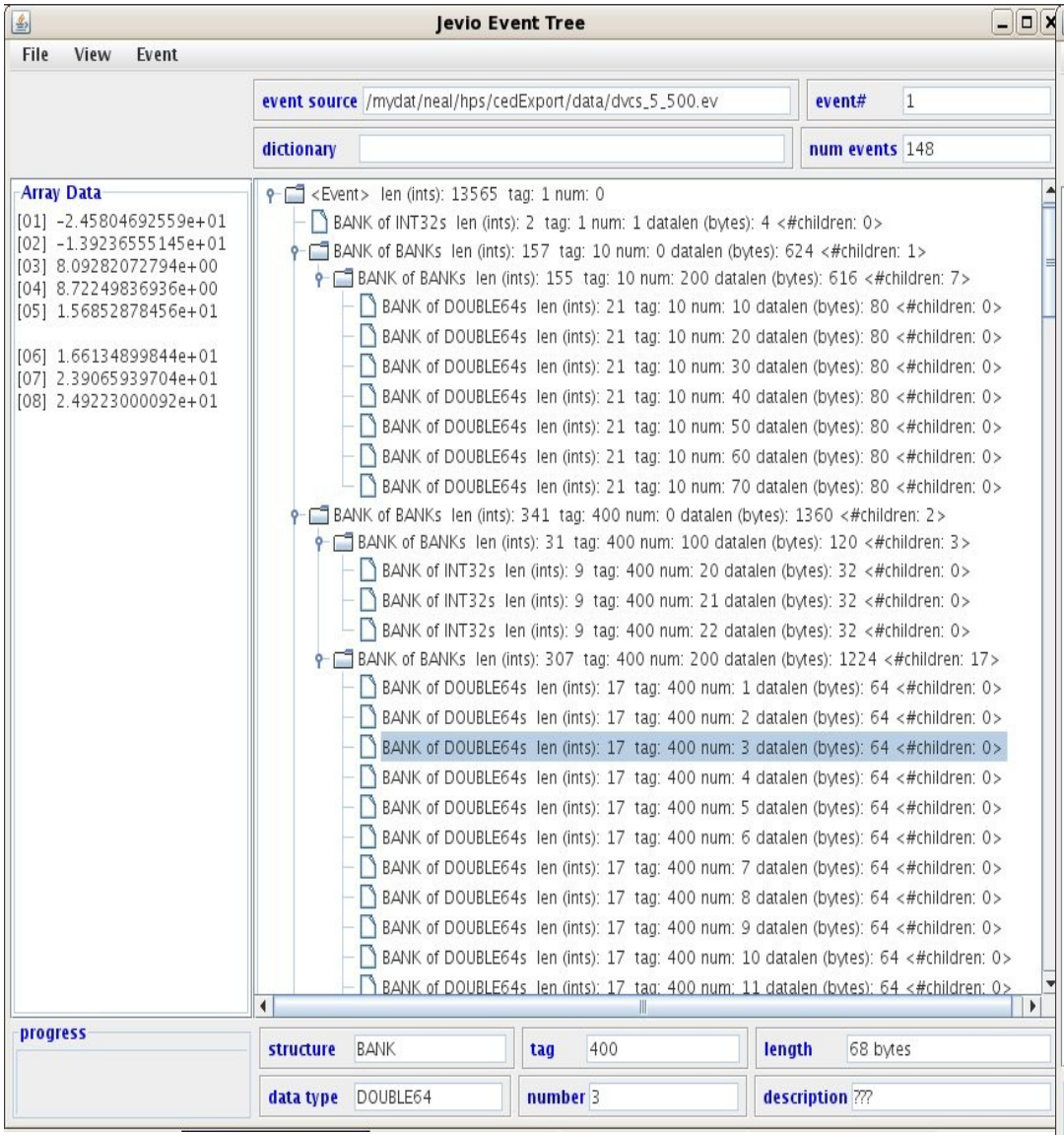
Bits	Value	Comment
31	1	This is a type defining word
30 – 27	2	Data type = event header
26 – 22	Slot ID	Set by VME64 backplane
21 – 20	Module type	0=FADC250, etc.
19 – 0	Trigger number	ADC processing chip #

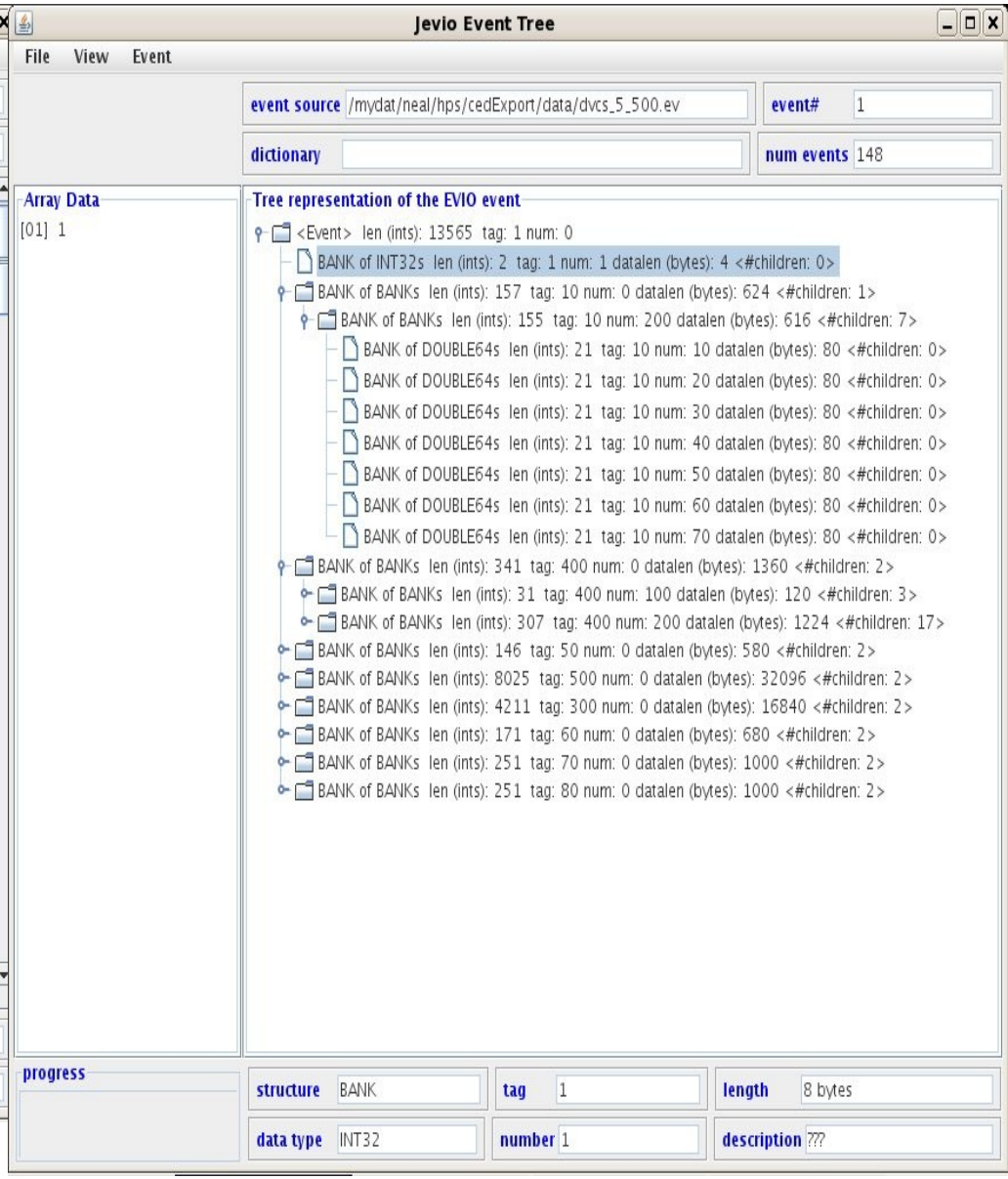
Examples from Class12 Event Display

```
import cnuphys.jevio.BaseStructureHeader;
import cnuphys.jevio.EventParser;
import cnuphys.jevio.EvioEvent;
import cnuphys.jevio.IEvioListener;
import cnuphys.jevio.IEvioStructure;

/**
 * This is the manager for CLAS specific events. This is where we
put the data into convenient arrays.
 * @author heddle
 *
 */
public class EventManager implements IEvioListener {

	//small number check
	private static final double TINY = 1.0e-10;

	/**
	 *  Maximum number of events we will allow for
accumulation
	 */
	public static final int MAXACCUMULATIONCOUNT =
10000;

	// singleton
	private static EventManager instance;

	// flag that set set to <code>true</code> if we are
accumulating events
	private boolean accumulating = false;

	// list of accumulated events
	private ArrayList<EvioEvent> _accumulatedEvents;
```

```
	/**
	 * Got a structure from the event source. This is where we look for
structures of interest and put
	 * them in conveniently accessible arrays.
	 * @param evioEvent the actual event.
	 * @param the structure received.
	 */
	@Override
	public void gotStructure(EvioEvent evioEvent, IEvioStructure
structure) {

		//grab the structures I'm interested in
		BaseStructureHeader header = structure.getHeader();
		int tag = header.getTag();
		int num = header.getNumber();

		// tag 500 has a lot of DC data
		if (tag == 500) {
			switch (num) {

			case 1:
				_dcGEMCArrayEdep = structure.getDoubleData();
				break;

			case 2:
				_dcGEMCArrayGlobalX =
structure.getDoubleData();
				break;
```

J[EVIO] Explorer





Summary

- Thanks to Sergey and Maurik we now have a general data description, example code for accessing evio event elements and sample evio data.
- Next step get an EVIO block available in LCIO
  - Then, demonstrate that we can access specific values in the blocks
- Develop set of access primitives for all elements and do consistency tests

Index to Documentation

stanford.edu https://confluence.slac.stanford.edu/display/hpsg/Event+Building+Scheme

Dashboard › Heavy Photon Search Group › ... › Data Definitions › Event Building Scheme

Browse Homer Neal Search

Edit Add Tools

Search

- Project Overview
- Meetings
- Research Opportunities
- Physics
- Beamline and Magnet
- Tracking and Vertexing Detector
- Calorimeter
- Muon Detector
- DAQ and Trigger
- Simulation and Reconstruction Software
- Conditions
- Data Definitions
  - ET
  - Event Building Scheme
  - EVIO Users Guide
  - Examples

Event Building Scheme

8 Added by Homer Neal, last edited by Homer Neal on Oct 11, 2011 (view change)

I attached event building scheme as it is presented by JLAB DAQ group at present time. It describes how data will travel through DAQ components up to ET (Event Transport) system. In the beginning of ET we can run some kind of translator to convert all data into more convenient format if needed.

So far our IO package (called EVIO) can handle banks with the same format (all shorts or all ints etc), but new version of EVIO will be able to have mixed-format banks described by user in FORTRAN style, like:

I,3F,6S,7(I,F)

and even have variable rows, for example format

I,2F,NS

means that the number of shorts in the end of data will be taken from data itself, so for given example you have to write data in following order:

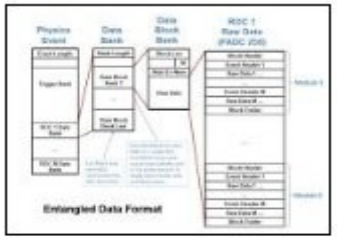
125(int), 1.1(float), 2.2(float), 2(int), 22(short), 33(short)

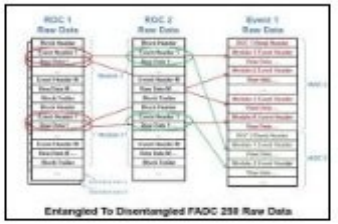
where 2 is the number of following shorts.

I also attached outdated EVIO manual, new one is not ready.

...

Regards, Sergey





- eventbuilding.pptx

Atlassian Confluence 3.2.1_01, the Enterprise Wiki: Intranet software for documentation and knowledge management | Report a bug | Atlassian News