ePIC+JANA2 Framework Needs and Timelines

Nathan Brei

Jefferson Lab

March 2, 2026

Concept

- This talk covers the open JANA2 and adjacent work packages that I hope to tackle over the next year
- I'm hoping to share:
  - What I see as the current ePIC+JANA pain points
  - My current plans for addressing these
- I'm hoping to collect:
  - Additional issues and requirements that are not on my radar
  - Knowledge about upcoming ePIC work that might intersect with my JANA2 work
  - Knowledge about ePIC deadlines that might impact scheduling and prioritization

Organization

1. Streaming
   - Timeframe splitter
   - Calibrations interface
   - Integrations with ERSAP and PanDA
2. AI/ML support
   - Multithreaded ONNX support
3. Heterogenous hardware
   - Training on a GPU
4. Ergonomics
   - Better support for enum JParameters
   - Better exception handling in JFactories
   - Wiring file support
   - Reducing the JMain boilerplate
   - Unifying the JANA2 and ePIC logger interfaces

Timeframe splitter

Priority: High

Current status

- Takuya's timeframe splitter works, but needs the latest version of JANA
- Latest version of JANA appears to do the wrong thing in the ePIC CI while handling an exception for one particular edge case: (https://github.com/JeffersonLab/JANA2/pull/487)
- Attempted to use the wiring file to cleanly separate the timeframe splitter configuration

Still needed

- Migrate Takuya's branch off of `jana::JOmniFactory`, which is going away now that the key functionality lives on `JFactory`
- Fix the bug and upgrade the JANA2 version used by EICrecon
- Create at least one timeframe-splitting test case

Customers and deadlines

- Takuya

Calibrations interface

Priority: High

Current status

- Previous work last year enabled us to handle calibration data symmetrically to physics event data, which will come in handy for online calibrations
- Declarative helpers such as `PodioInput<>` are now usable inside `JFactories`, `JEventUnfolders`, `JEventSources`, etc

Next steps

- The `Resource<>` helper (analogous to the `PodioInput<>` helper) needs further design and development. The current version reduces boilerplate but doesn't do the right thing under the hood yet

Customers and deadlines

- JLab's CEAC project is experimenting with online calibrations for GlueX. GlueX uses the old-style, JService-based JCalibrationManager, and this is a major obstacle for them.

ERSAP/PanDAS integration

Priority: Medium

Current status

- JANA2 is designed to be mostly independent of ERSAP and PanDAS
- A persistently-running event source that watches a directory was created and demonstrated, but hasn't been merged. https://github.com/eic/EICrecon/tree/nbrei_file_streaming

Next steps

- The official example of how to integrate JANA2 inside an ERSAP microservice is out-of-date. Nobody has used `JEventFolders` yet. These would be an excellent replacement for `JEventGroups`.
- JANA2's status reporting tools aren't designed for persistent, cloud-based workflows. Having a JANA2-specific Grafana widget would be nice.

Customers and deadlines

- A number of smaller projects around JLab are using ERSAP and JANA2 together, including an LDRD project of mine

Multithreaded ONNX support

Priority: High

Current status

- ONNX is used in EICrecon, but only in single-threaded mode

Next steps

- Figure out the best way to handle this, and communicate with users
- Create an example as part of the JANA tutorial

Customers and deadlines

- This exact functionality is also requested by Casey at JLab

Model training on GPU

Priority: Low

Current status

- Models are trained outside of JANA2
- Model training on a GPU is fundamentally a critical section, which limits the performance benefits of doing this from inside JANA2
- Podio data is laid out in rows, not columns, which makes this harder
- Offloading a JFactory's computation and subsequently running other JFactories on the results constitutes a cycle in the underlying arrow topology, which makes this harder

Next steps

- I can't solve the abstract problem until I've solved at least one concrete instance
- A performance analysis of training from Podio vs RDataFrames, inside a JEventProcessor vs outside JANA

Enum JParameters

Priority: Medium

Current status

- JParameters have no special support for enums, so each enum parameter has its own ad-hoc, inconsistent validation, parsing, serialization, and documentation: https://github.com/JeffersonLab/JANA2/issues/318

Next steps

- Create `JEnumParameter` and `EnumParameter<T>` subclasses
- Gradually adopt

Customers and Deadlines

- Sebouh
- Me

Improve exception handling of JFactories

Priority: Medium

Current status

- Some factories throw exceptions, particularly with weirder detector configurations. It is apparently **not** sufficient to say "return empty collections instead" or "stop all processing when an exception occurs". We need to simultaneously:
  - Continue processing the factories in the chain
  - Let downstream know that a failure occurred
  - Provide an empty collection anyway so that downstream doesn't segfault
- This is been a headache for a while:
  - https://github.com/eic/EICrecon/pull/1069 (Oct 11, 2023)
  - https://github.com/eic/EICrecon/pull/2357 (Jan 22, 2026)

Next steps

- Fix my latest bug (https://github.com/JeffersonLab/JANA2/pull/487)
- Design a better mechanism (probably requires collaboration)

Wiring file support

Priority: Medium

Current status

- JANA2 v2026.01.00 onwards supports specifying configuration values via a wiring file
  - Set parameter values and additional plugins to load
  - Enable or disable entire components, such as TimeframeSplitter
  - Override JOmniFactoryGenerator input/output collection names and parameter values
  - Does **not** support inheriting from other wiring files at this time

Next steps

- Use the wiring file for timeframe splitting (Note this is a separate task from the timeframe splitting deliverable)
- See how much of the eicrecon CLI can be expressed inside the wiring file itself

Customers and Deadlines

- None

JMain boilerplate reduction

Priority: Low

Current status

- The division of responsibility between JANA2 and EICrecon is unintuitively blurred between `eicrecon.cc`, `eicrecon_cli.cc`, `JMain.h`, `JBenchmarker.h`, and `JSignalHandler.h`.
- EICrecon reimplements functionality (accessing JANA2 internals) it shouldn't have to.
- The overwhelming majority of things that a custom `main()` function needs to do can also be done via the wiring file

Next steps

- New JANA2 projects are using a wiring file instead of a custom `main()`.

Unified logger interfaces

Priority: Low

Current status

- JANA2 logger and EICrecon logger are independent
- Most JANA2 logging is disabled by default in EICrecon, but not everything, and the two streams are awkwardly merged and formatted inconsistently
- JANA2 provides named logging groups. JANA2 internals are all assigned a single logging group, whereas individual JFactories, etc, get their own.
- Relationship between JANA2 factory logging and EICrecon algorithm logging is confusing

Next steps

- Provide a mechanism like ACTS, where JANA's logger can be redirected to the user's logger of choice? https://github.com/JeffersonLab/JANA2/issues/250

Customers and Deadlines

- None

Summary

Theme	Work package	Priority	Customers	Deadlines
Streaming	**Timeframe splitter**	**High**	Takuya	
	Calibrations interface	**High**	CEAC, ePIC SRO?	
	ERSAP+PanDAS integration	Med		
AI/ML	**Multithreaded inference with ONNX**	**High**		
HetHW	Training on GPU	Low		
Ergonomics	**Support enum JParameters**	**Med**	Sebouh	
	Improve exception handling	Med		
	Wiring file support	Med		
	Reduce JMain boilerplate	Low		
	Unified loggers	Low		

Thank you!