Provenance update talk

https://indico.in2p3.fr/event/23481/contributions/94063

1. Defining the content of a **minimum provenance (last-step provenance)**
   - List of keywords related to the last activity and context
2. **Serializing** provenance
   - Both **human** and **machine** readable
   - Explore W3C formats, and YAML / VOTABLE / VOEvent formats
3. Provenance and **workflows**
4. From provenance "**on-top**" to provenance "**inside**"
   - How to map specific provenance information into the IVOA model?
   - How to introduce provenance **capture** inside a pipeline?
5. Provenance **storage**
6. Provenance **exploration** and **visualization**
   - Access protocols (ProvTAP, ProvSAP)
   - `voprov` Python package, or other tools

Provenance hack-a-thon summary

last-step provenance

- List of attributes **embedded** in the entity,
- **Simplified** view of the **full** provenance
- Issues when relying on embedded provenance only : if files are modified by different activities during the process, they cannot carry this last-step provenance...
- Full provenance should thus be stored **externally** (prov file, log file or database)
- In a dedicated database, provenance information can then be queried (prototypes by CTA and CDS)

Access to provenance

- ProvSAP : simple access protocol, gives the graph for an identifier (entity or activity)
- ProvTAP : full access to provenance in a DB, + view on last-step provenance table

Provenance hack-a-thon summary

Capture detailed provenance from inside the code

- example of **logprov** to decorate Python scripts, data analysis use case
- Pipeline framework (e.g. ctapipe), data preparation use case

Points to clarify

- "**Provenance inside**" refers to where the provenance recording is performed, while "**embedded**" provenance is provenance information carried by an entity
- **Reusability** and **Reproducibility** : different requirements and solutions