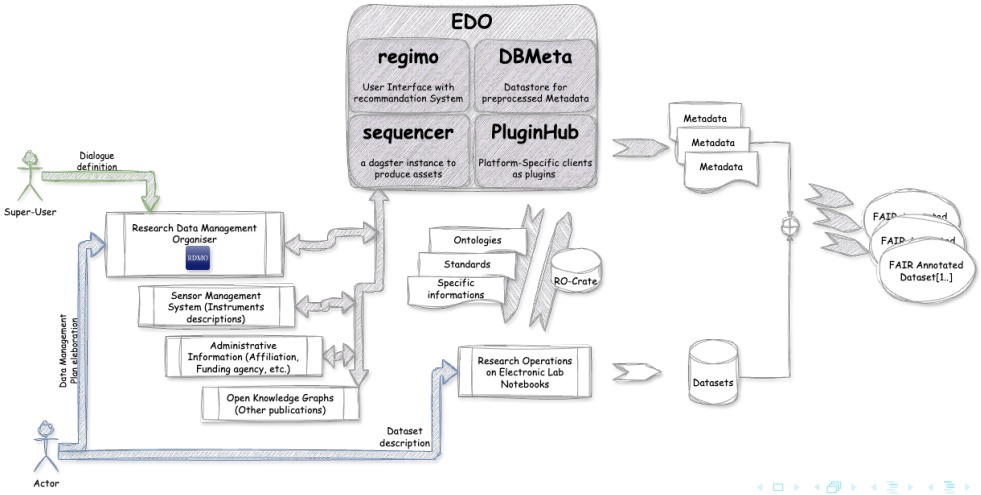
Effective use of PIDINST in the automation of data sets publication

Verifiable Research Objects using RO-Crate and PROV-O

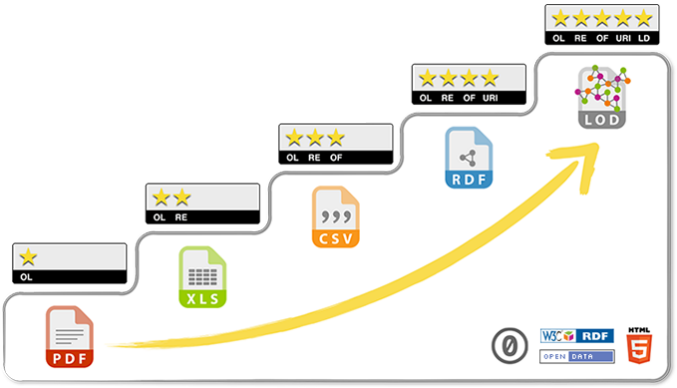
A. Koubaa

May 6, 2026

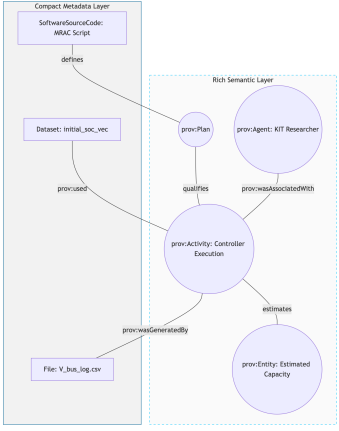
EDO Overview



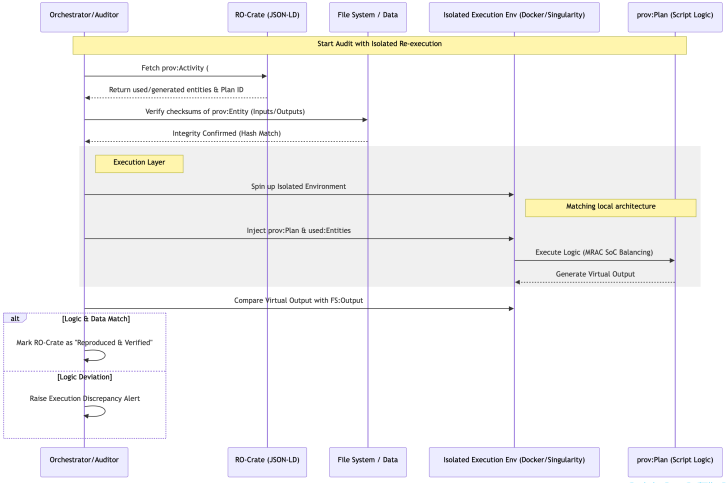
Motivation


OL
PDF
OL RE
XLS
OL RE OF
CSV
OL RE OF URI
RDF
OL RE OF URI LD
LOD
W3C RDF
OPEN DATA
HTML
5

The Vision


Compact Metadata Layer
SoftwareSourceCode: MRAC Script
defines
Dataset: initial_soc_vec
prov:used
File: V_bus_log.csv
Rich Semantic Layer
prov:Plan
prov:Agent: KIT Researcher
qualifies
prov:wasAssociatedWith
prov:Activity: Controller Execution
estimates
prov:wasGeneratedBy
prov:Entity: Estimated Capacity

The Vision


Orchestrator/Auditor
RO-Crate (JSON-LD)
File System / Data
Isolated Execution Env (Docker/Singularity)
prov:Plan (Script Logic)
Start Audit with Isolated Re-execution
Fetch prov:Activity (
Return used/generated entities & Plan ID
Verify checksums of prov:Entity (Inputs/Outputs)
Integrity Confirmed (Hash Match)
Execution Layer
Spin up Isolated Environment
Matching local architecture
Inject prov:Plan & used:Entities
Execute Logic (MRAC SoC Balancing)
Generate Virtual Output
Compare Virtual Output with FS:Output
alt
[Logic & Data Match]
Mark RO-Crate as "Reproduced & Verified"
[Logic Deviation]
Raise Execution Discrepancy Alert
Orchestrator/Auditor
RO-Crate (JSON-LD)
File System / Data
Isolated Execution Env (Docker/Singularity)
prov:Plan (Script Logic)

The Provenance Audit Framework

Structural Verification of Executable Research Objects

Discovery & Integrity Layers

- Resolves `prov:Activity` from RO-Crate metadata to establish context.
- Executes physical checksum verification (e.g., `V_bus_log.csv`) against metadata hashes to ensure data immutability.

Logical & Temporal Layers

- Validates `prov:hadPlan` to verify the `Est` function's adaptive control logic.
- Uses `prov:startedAtTime` and `prov:endedAtTime` to confirm the $T_{sim} = 2400$s window.

Goal: Certify execution as a deterministic result of logic applied to data.

The Isolated Execution Layer

Guaranteeing Redoability through Architectural Parity

Trusted Execution Environment (TEE)

Instantiates a containerised sandbox.

Reproduction Workflow

1. **Injection:** Permits only `prov:used` entities to eliminate bias.
2. **Reproduction:** Triggers `prov:Plan` for MRAC adaptation.
3. **Validation:** Cross-references outputs using tolerance ϵ.

Scientific Impact

- Ensures estimated plant parameters converge consistently.
- Validates total storage capacity estimation across environments.

Integrating RO-Crate and PROV-O

- **RO-Crate:** A community-driven method for packaging research data with structured metadata (JSON-LD).
- **PROV-O:** The W3C Provenance Ontology providing a model to map the origins of data.
- **The Goal:** Use RO-Crate to store the "What" (data/tools) and PROV-O to describe the "How" (process/execution).

Core PROV-O Relationships

- `prov:Entity`: Data objects or software within the crate.
- `prov:Activity`: The computation or manual step performed.
- `prov:Agent`: The researcher or system responsible.

Linking Entities for Traceability

To make operations verifiable, the RO-Crate `ro-crate-metadata.json` must link files to their execution context:

1. **wasGeneratedBy**: Links a result file (*Entity*) to a specific script run (*Activity*).
2. **used**: Links the activity to its input datasets and parameter files.
3. **wasAssociatedWith**: Connects the activity to the *Agent* (e.g., a KIT research fellow) or a software environment.

Benefit: This graph allows a machine to traverse from a plot back to the exact version of the raw data and code used.

Enabling Re-runnability and Parameterisation

- **Verifiability:** Independent auditors can check if the `prov:Activity` outputs match the provided `prov:Entity`.
- **Redoability:** By describing the `SoftwareApplication` as a `prov:Plan`, others can swap input entities.
- **Parameter Tracking:** Store configuration files as `prov:Entity` linked via the `used` property.

Workflow Logic

If a researcher wants to test a new parameterisation, they identify the `Activity` in the RO-Crate, substitute the input `Entity`, and re-execute the linked `SoftwareSourceCode`.

PIDINST: Persistent Identification of Instruments

Anchoring Hardware Metadata in PROV-O

The Role of PIDINST

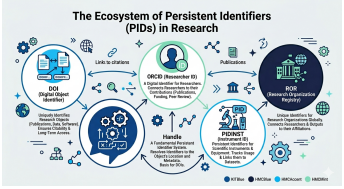
- Assigns a **Persistent Identifier (DOI/Handle)** to physical instruments (e.g., Battery Cyclers, Sensors).
- Provides machine-readable metadata: Manufacturer, Model, Serial Number, and Calibration History.

PROV-O Integration

- **prov:Agent:** The instrument acts as an automated agent performing an activity.
- **prov:Entity:** Represents the instrument state at the time of simulation.
- Enables the Auditor to verify if the hardware used (e.g., in Karlsruhe) matches the specifications required by the `prov:Plan`.

The Persistent Identifier (PID) Ecosystem

Interconnected Metadata for Verifiable Research



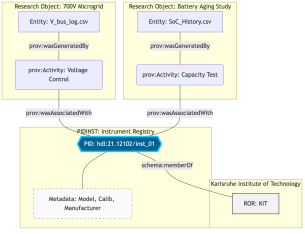
Core Infrastructure:

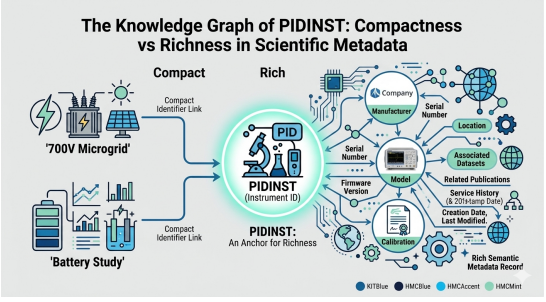
- **ORCID/ROR:** Link researchers and their affiliations.
- **DOI/Handle:** Ensure long-term citability of datasets and software.

Experimental Context:

- **PIDINST:** Connects physical instruments to the `prov:Activity`.
- **Graph Utility:** Enables a reachness of information across the audit chain.

PIDINST


Research Object: 700V Microgrid
Entity: V_bus_log.csv
prov:wasGeneratedBy
prov:Activity: Voltage Control
prov:wasAssociatedWith
Research Object: Battery Aging Study
Entity: SoC_History.csv
prov:wasGeneratedBy
prov:Activity: Capacity Test
prov:wasAssociatedWith
PIDINST: Instrument Registry
PID: hdl:21.12102/inst_01
schema:memberOf
Karlsruhe Institute of Technology
ROR: KIT
Metadata: Model, Calib, Manufacturer


The Knowledge Graph of PIDINST: Compactness vs Richness in Scientific Metadata
Compact
Rich
'700V Microgrid'
'Battery Study'
Compact Identifier Link
Compact Identifier Link
PID
PIDINST
(Instrument ID)
PIDINST:
An Anchor for Richness
Company
Manufacturer
Serial Number
Location
Serial Number
Associated Datasets
Model
Related Publications
Firmware Version
Service History (& 20ts+amp Date)
Creation Date, Last Modified.
Calibration
Rich Semantic Metadata Record
KITBlue
HMCBlue
HMCAccent
HMCMint

1. Structural Compactness vs. Density

Graph Density (D)

For a graph with V vertices and E edges, density represents the ratio of actual connections to potential ones:

$$D = \frac{|E|}{|V|(|V| - 1)}$$

- **The PIDINST Effect:** By substituting internal metadata clusters (redundant nodes for serial numbers, models, and calibration) with a single URI, we reduce $|E|$ and $|V|$ locally.
- **Theorem:** A more compact graph minimizes D while maintaining the *Reachability* of information.

Question: At what point does a graph become too sparse to be self-describing?

2. Semantic Entropy and Richness

Graph Entropy (H)

Based on Shannon's theory, we measure the complexity of the semantic distribution:

$$H(G) = -\sum_{i=1}^{n} P(\text{rel}_i) \log_b P(\text{rel}_i)$$

- **Richness Factor:** High richness is defined by a diverse set of PROV-O relations (`used`, `wasInformedBy`, `hadPlan`).
- **Optimization:** Using PIDs increases $H(G)$ per node, as each node carries higher "Significance" within the network topology.

Discussion: Does increasing entropy always improve reproducibility?

3. Centrality and The "Bridge" Effect

Betweenness Centrality (C_B)

Measures the extent to which a node lies on paths between others:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

- **PIDINST as a Hub:** In an institutional graph at KIT, PIDINST identifiers exhibit the highest C_B, acting as the "glue" between disparate Research Objects.
- **Verification Efficiency:** The Auditor/Orchestrator can reach the context or the *MRAC* logic faster by traversing these high-centrality nodes.

Question: Does reliance on central hubs create a single point of failure for EDO?

4. Discussion: The Efficiency of Truth

Towards a Unified Metric for Executable Research Objects

- **The Compactness-Richness Paradox:** We seek to minimize the "Data Footprint" of the RO-Crate while maximizing its "Logical Depth."
- **Proposed Discussion Points:**
  1. Can we define a **Reproducibility Index** based on the average path length between a result and its PIDINST/Plan?
  2. How does architectural parity shift the graph from a *Descriptive* state to an *Executable* state?
  3. Should "FAIRness" be measured by the mathematical modularity of the institutional Knowledge Graph?

Opening for Group Discussion

Mathematical Modularity (Q) in the KG

Quantifying the Strength of Semantic Clusters

The Modularity Formula

Modularity (Q) measures how much more "clustered" a graph is compared to a random distribution of edges:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

- A_{ij}: Relationship between nodes i and j.
- k: Degree of nodes (number of connections).
- $\delta(c_i, c_j)$: Limits the sum to nodes within the same community (e.g., your specific 700V microgrid simulation).

High Q indicates a "healthy" graph where data is logically compartmentalized

Structural Impact: PIDs vs. Redundancy

How PIDINST Protects Graph Modularity

Without PIDINST

- Metadata is duplicated across every experiment.
- **Result:** "Fuzzy" communities and lower Q.
- The graph becomes a "Big Ball of Mud."

With PIDINST

- The instrument is a central hub *between* clusters.
- **Result:** High internal density with strategic external links.
- Maximises Q for the institutional graph.

The Reachness Paradox

We achieve higher **Reachness** (access to rich sensor data) while maintaining lower **Graph Density** within the Research Object.

Modularity in the Redoability Audit

Ensuring Scalability for the KIT Infrastructure

High modularity is essential for the **Orchestrator** when verifying an Executable Research Object (ERO):

Isolation: The Auditor can extract the 700V microgrid provenance chain without "leakage" from unrelated battery aging studies.

Search Efficiency: Computational cost of pathfinding (from result to `prov:Plan`) remains low even as the Knowledge Graph grows.

Validation: High Q allows for automated consistency checks of the **MRAC** logic against the **Darwin ARM64** environment variables.

Discussion: Should Modularity be an automated KPI for "FAIR" institutional data management?