OGC® DOCUMENT: 24-036

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t20-D022

Open Geospatial Consortium

OGC TESTBED 20 GDC PROVENANCE DEMONSTRATION REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2025-02-13
Approval Date: 2025-06-12
Publication Date: 2025-06-23
Editor: Jonas Eberle

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogc, testbed, provenance, geo data cube, processes, OpenEO, STAC, W3C PROV, metadata

II CONTRIBUTORS

All questions regarding this document should be directed to the editor or the contributors:

Table — Contributors

NAME	ORGANIZATION	ROLE
Jonas Eberle	German Aerospace Center	Editor
Michele Claus	Eurac Research	Contributor
Gérald Fenoy	GeoLabs	Contributor
Francis Charette Migneault	CRIM	Contributor

III OVERVIEW

The goal of the OGC Testbed 20 GeoDataCube (GDC) Provenance Demonstrator was to establish a standardized approach for exchanging metadata and provenance information in geospatial workflows. By utilizing the GDC API Profiles, this Testbed task demonstrated how transparency and reproducibility can be ensured, enabling users to access essential details about data sources and processing methods. This is particularly beneficial for both open science initiatives and restricted workflows, where traceability and accountability are crucial.

A primary objective of this Testbed 20 task was to enhance the draft GDC API Profiles specification to enable seamless metadata and provenance exchange for GeoDataCubes. The work focused on improving interoperability by aligning with established metadata standards and enhancing the tracking of workflow identifiers, metadata definitions, and remote process chains. Additionally, the task participants aimed to refine STAC metadata to improve transparency in geospatial data processing.

Supporting open science was another key objective, achieved through machine-readable outputs. Through these efforts, the GDC Provenance Demonstrator enhances the reproducibility of both scientific and restricted workflows while providing a reliable and standardized framework for geospatial data management.

IV FUTURE OUTLOOK

While W3C PROV-compliant metadata integration into OGC API — Processes — Part 4: Job Management implementations have proven effective for provenance tracking, the Processes API Standard requires clearer guidelines for consistent implementation. Key improvements include aligning jobID and runID in workflows, defining essential metadata, and improving provenance tracking for remote process chains.

Enhancing interoperability between OGC API — Processes implementations and other geospatial frameworks could enable seamless process chaining. Additionally, STAC Items should incorporate provenance metadata for transparency, with the scientific citation STAC extension supporting reproducibility. Refinements to the STAC processing extension, such as `derived_from` links, would further improve dataset traceability.

V VALUE PROPOSITION

The GDC Provenance demonstrator offers significant value to various stakeholders:

- For Researchers: Provides transparent and reproducible data workflows.
- For Data Providers: Facilitates standardized metadata and provenance sharing, enhancing the discoverability and usability of their datasets.
- For Policy Makers: Supports evidence-based decision-making by providing traceable and credible data sources.
- For Developers: Offers an API framework that can be integrated into diverse applications.

The ability to balance transparency and privacy ensures its relevance in both open and restricted contexts, maximizing its usefulness.

1 INTRODUCTION

1 INTRODUCTION

The GDC Provenance Demonstrator was developed to facilitate the exchange of information on data sources (i.e., metadata) and provenance information (i.e., processing steps, algorithms, specifications) for a GeoDataCube in both open science and restricted workflows.

GeoDataCubes are integral to modern geospatial data analysis, offering a structured approach to managing multidimensional geospatial datasets. However, the effective use of data cubes often depends on accessible and standardized metadata and provenance information. Therefore, the goal of this Testbed-20 project was to develop a demonstrator for the GDC API specification that addresses these needs—ensuring that GeoDataCubes can be effectively utilized in both open science and restricted workflows.

1.1. Aims

Develop a GDC Provenance demonstrator that enables exchanges of information on data sources (i.e. metadata) and provenance (i.e. processing steps, algorithms, specifications) for a given GeoDataCube, and consider its use in open science workflows and workflows where metadata and provenance cannot be fully disclosed.

The primary aim of the project is to design and implement a GDC Provenance demonstrator that:

- Facilitates Information Exchange: Enables seamless access to metadata and provenance information.
- Supports Diverse Workflows: Caters to both open science and restricted environments.
- Promotes Standards and Interoperability: Adheres to established standards for metadata and provenance management.

1.2. Objectives

To achieve the stated goal, the project focuses on the following objectives:

- **Develop API Functionality:** Implement core features for querying and retrieving metadata and provenance information.
- **Ensure Standards Compliance:** Align with metadata and provenance standards.

- **Enable Open Science Integration:** Support machine-readable outputs and interoperability with platforms such as Jupyter Notebooks and Zenodo.
- **Demonstrate Practical Applications:** Develop use cases to showcase the API's functionality in real-world scenarios.

By focusing on these objectives, the project aims to create an API that addresses current limitations and supports future advancements in geospatial data management.

2 TOPICS

2 TOPICS

2.1. Terminology

To ensure clarity and consistency, the following terms are used throughout this report:

- GeoDataCube (GDC): A structured multidimensional representation of geospatial data, often organized by spatial, temporal, and thematic dimensions.
- Metadata: Descriptive information about data, including details about its source, structure, acquisition methods, spatial/temporal extent, and quality.
- Provenance: Information detailing the processes, algorithms, and transformations applied to data, enabling traceability and reproducibility.
- Open Science Workflows: Scientific workflows designed to promote transparency, reproducibility, and collaboration by openly sharing data, methods, and results.
- W3C PROV: TA group of standards developed by the World Wide Web Consortium (W3C) for representing provenance information.

2.2. API-based provenance

This section outlines the available resource paths of the Provenance Demo API related to metadata within the job execution framework. These endpoints provide structured information about the execution, inputs, outputs, and actors involved in a computational job or workflow. They support tracking, auditing, and reproducibility by exposing key metadata about each job and its execution steps.

The following table details the specific resource paths and their respective functionalities:

Table 1

ENDPOINT	DESCRIPTION
`/jobs/{jobID}/prov/info`	Metadata about the Research Object packaging information.
`/jobs/{jobID}/prov/who`	Metadata about *who* ran the Job, typically as an ORCID.

ENDPOINT	DESCRIPTION
`/jobs/{jobID}/prov/runs`	Obtain the list of `runID` steps of the Workflow within the Job.
`/jobs/{jobID}/prov/run`	Metadata of the main Job and any nested step runs in the case of a Workflow.
`/jobs/{jobID}/prov/inputs`	Metadata about the Job input IDs.
`/jobs/{jobID}/prov/outputs`	Metadata about the Job output IDs.
`/jobs/{jobID}/prov/[run|inputs|outputs]/{runID}`	Same as their respective definitions above, but for a specific step of a Workflow, as applicable.

For all of those endpoints, the returned contents are in `text/plain`, although some alternate representations could probably be defined. For example, calling the `GET /jobs/{jobID}/prov/info` operation would result in the following.

```
Research Object of CWL workflow run
Research Object ID: arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/
Profile: https://w3id.org/cwl/prov/0.6.0
Workflow run ID: urn:uuid:6617474d-4dc1-4f77-a993-2527c2618a7c
Packaged: 2024-12-21
```

Listing 1 — Job Provenance Research Object Packaging Information

The `runID` combinations can be employed for cases where a workflow involving multiple steps is employed. At the very least, there should be exactly one `runID` matching the main step of the `jobID` corresponding to the top-most process execution. All other `runID` could represent either a distinct `jobID` if the server chose to represent nested execution steps as distinct queryable jobs, or any other UUID for internal reference. This `runID` allows the underlying workflow engine to record individual processes in a chain with the respective execution metadata, their intermediate inputs, and resulting outputs. Using those `runID`, the provenance information can also better indicate the sequence of operations of each step, by attributing metadata such as how the inputs/outputs where chained between them, or by providing start/finish date-time details.

For example, the following contents correspond to the `GET /jobs/{jobID}/prov/run` response. Since the `echo` process used in this case is only an atomic operation, only a single `GET /jobs/{jobID}/prov/run/{runID}` applies, and yields the same contents, with `jobID = runID = 6617474d-4dc1-4f77-a993-2527c2618a7c`.

```
2024-12-21 05:46:04.879859                              Flow 6617474d-4dc1-4f77-
a993-2527c2618a7c [ Job Information
2024-12-21 05:46:04.883022                              In   2ef7bde608ce5404e97d5f
042f95f89f1c232871 < wf:main/message
2024-12-21 05:46:04.940812                              In   2ef7bde608ce5404e97d5f
042f95f89f1c232871 < wf:main/echo/message
                         2024-12-21 05:46:06.207918 Out  c2ffa0b6-271a-4a07-
8070-9ba6db93893e > wf:main/echo/output
                         2024-12-21 05:46:06.207918 Out  16789964-9acd-4a91-
b636-5ee7eca20011 > wf:main/echo/PACKAGE_OUTPUT_HOOK_LOG_b563c5f1-d4e0-46f0-b5ea-
6b20b8aff93a
                         2024-12-21 05:46:06.207918 Out  6cad4e06-b51d-442d-
b480-9e0199513ef7 > wf:main/echo/stdout.log
```

```
                          2024-12-21 05:46:06.211805 Out  c2ffa0b6-271a-4a07-
8070-9ba6db93893e > wf:main/primary/output
                          2024-12-21 05:46:06.211805 Out  16789964-9acd-4a91-
b636-5ee7eca20011 > wf:main/primary/PACKAGE_OUTPUT_HOOK_LOG_b563c5f1-d4e0-46f0-
b5ea-6b20b8aff93a
                          2024-12-21 05:46:06.211805 Out  6cad4e06-b51d-442d-
b480-9e0199513ef7 > wf:main/primary/stdout.log
                          2024-12-21 05:46:06.207897 Flow 6617474d-4dc1-4f77-
a993-2527c2618a7c ] Job Information  (0:00:01.328038)
```

Listing 2 — Job Provenance Run Information

2.3. STAC-based provenance

Provenance tracking involves documenting the origins, transformations, and methods associated with data, ensuring transparency and reproducibility. STAC, with its structured JSON-based metadata format, offers a standardized way to organize and describe geospatial assets, while the STAC Processing Extension enhances this capability by capturing significant details about processing workflows.

The processing extension adds fields to a STAC Item's metadata, typically under a processing object:

- `processing:software`: This field captures the tools and libraries used during the process execution. The various software used, along with their versions as well as the URL to the source code, can be explicitly recorded. This ensures that users and systems can trace back the exact configurations and tools used to generate the data.
- `processing:expression`: This field records the specific mathematical or computational expressions applied during data processing. This includes formulas, algorithms, or workflows that transform input data into output results. By documenting expressions, the back end provides a clear representation of how results were derived, further enhancing transparency and reproducibility.
- `processing:facility`: This field represents the facility that produced the data. It could be particularly useful to trace the exact facility in a federated cloud environment such as the openEO Platform.
- `processing:datetime`: This field is used to log the date and time of each processing task. This information is valuable when working with cloud infrastructures using queueing systems, which do not allow knowing in advance when processing will start.

2.3.1. Aligning with the FAIR Principles

The implementation of STAC and the Processing Extension directly supports the FAIR principles, promoting better data management and sharing.

1. Findable: STAC catalogs, collections, and items are inherently designed to enhance discoverability. By indexing batch job results in a STAC-compatible

format, the Eurac Research implementation ensures that datasets are well-organized and easily searchable. Metadata fields like spatial and temporal extents, combined with detailed processing information, provide clear entry points for users to locate relevant data.

2. Accessible: The use of open standards such as STAC and openEO APIs ensures interoperability with a wide range of tools and platforms. By linking metadata to data assets via URLs and adhering to common formats, the back end makes batch job results readily accessible to both humans and machines.

3. Interoperable: STAC's JSON-based structure and the Processing Extension's standardized fields enable seamless integration across diverse systems. The metadata aligns with widely recognized geospatial conventions, facilitating interoperability with other data catalogs, visualization tools, and analysis pipelines.

4. Reusable: The detailed provenance information captured through the Processing Extension of STAC ensures that data can be reused with confidence. Users can understand the lineage, processing methods, and quality of the data, which is crucial for reproducibility and meaningful application in new contexts. Moreover, adherence to open standards maximizes the longevity and utility of the datasets.

3 OUTLOOK

3 OUTLOOK

The integration of W3C PROV-compliant metadata within an implementation of OGC API — Processes — Part 4: Job Management has demonstrated that it is feasible to effectively provide provenance information in the context of OGC API Standards. However, the OGC API Processes standard currently lacks detailed guidance on how provenance should be consistently applied across different implementations.

One key area for improvement is the alignment of jobID and runID in nested workflow executions, as well as their consistency with the UUID assigned to the Research Object. Additionally, essential metadata related to the server instance, reference implementation, and other contextual details remain undefined, making provenance interpretation challenging. To enhance data integrity and trust in GDC results, OGC should establish clearer guidelines for implementing PROV in OGC standards and their implementations to ensure consistent and interoperable data provenance.

Another important aspect is the tracking of provenance in remote process chains. The OGC API — Processes — Part 4: Job Management draft standard should provide a structured approach for documenting the provenance of distributed workflows, particularly in cases involving multiple processing engines and server instances. The current lack of clear guidance in this area limits transparency and reproducibility.

Interoperability between OGC API — Processes and other geospatial processing frameworks also presents an opportunity for further exploration. If an OGC Application Package with a single output can be described in a way that enables seamless integration with other workflow engines, it could facilitate process chaining across platforms. However, this would require support for defining process graphs that span multiple server instances.

Additionally, STAC items generated from implementations of the OGC API Processes standard would benefit from improved provenance tracking. When provenance tracking is enabled at job initiation, the resulting STAC items should include associated metadata to enhance transparency. Furthermore, in cases where reproducibility is crucial, the Scientific Citation STAC Extension should be utilized based on metadata from standardized geospatial workflows.

Further refinements should also be made to the STAC processing extension, particularly by integrating additional metadata fields such as derived_from links. These links would improve traceability by associating processed datasets with their original sources.

By addressing these recommendations, the OGC community can establish a more robust and standardized approach to provenance tracking, ensuring better interoperability, transparency, and reliability in geospatial data workflows.

4

SECURITY, PRIVACY AND ETHICAL CONSIDERATIONS

4 SECURITY, PRIVACY AND ETHICAL CONSIDERATIONS

During the course of this project, a thorough review was conducted to identify any potential security, privacy, and ethical concerns. After careful evaluation, it was determined that none of these considerations were relevant to the scope and nature of this project. Therefore, no specific measures or actions were required in these areas.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Common Workflow Language Project (2024). cwltool: CWL reference implementation. Retrieved from https://github.com/common-workflow-language/cwltool

[2] Paul Groth, Luc Moreau, editors (2013). PROV-Overview: An Overview of the PROV Family of Documents. W3C Working Group Note. URL: https://www.w3.org/TR/prov-overview/

[3] Luc Moreau, Paolo Missier, editors (2013). PROV-DM: The PROV Data Model. W3C Recommendation. URL: http://www.w3.org/TR/2013/REC-prov-dm-20130430/

[4] Farah Zaib Khan, Stian Soiland-Reyes, Richard O Sinnott, Andrew Lonie, Carole Goble, Michael R Crusoe (2019): Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. GigaScience 8(11):giz095 https://doi.org/10.1093/gigascience/giz095

[5] Pedro Gonçalves, editor (2021) 'OGC Best Practice for Earth Observation Application Package', http://www.opengis.net/doc/BP/eoap/1.0.

[6] Benjamin Pross, Panagiotis A. Vretanos, editors (2021). 'OGC 18-062r2: OGC API — Processes — Part 1: Core', http://www.opengis.net/doc/IS/ogcapi-processes-1/1.0.

[7] STAC Community (2024). SpatioTemporal Asset Catalog (STAC) Specification. Retrieved from https://stacspec.org

[8] Francis Charette-Migneault (2025). crim-ca/weaver:6.2.0 (6.2.0). Zenodo. https://doi.org/10.5281/zenodo.14826363

[9] Gérald Fenoy, editors (2025). 'DRAFT OGC API — Processes — Part 4: Job Management', https://docs.ogc.org/DRAFTS/24-051.html.

A

ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS

ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS

API	Application Programming Interface
CWL	Common Workflow Language
DRU	Deploy, Replace, Undeploy
GDC	Geo Data Cube
PROV	Provenance
STAC	Spatio Temporal Asset Catalog
W3C	World Wide Web Consortium

B

ANNEX B (INFORMATIVE) EURAC RESEARCH GDC PROVENANCE DEMO

ANNEX B (INFORMATIVE) EURAC RESEARCH GDC PROVENANCE DEMO

B.1. Using STAC metadata for provenance

The integration of SpatioTemporal Asset Catalog (STAC) standards with the Processing Extension in Eurac Researchs' openEO back-end provides a robust framework for indexing and managing the results of batch job processing. This implementation follows best practices for geospatial data management, particularly in the realms of provenance tracking and adherence to the FAIR (Findable, Accessible, Interoperable, and Reusable) principles.

Here are the fields implemented and available at Eurac Research:

- `processing:software` to capture the tools and libraries used during the openEO batch job execution. Example:

```
{
  "processing:software": [
    {
      "name": "openeo-spring-driver",
      "version": "1.2.0",
      "url": "https://github.com/Open-EO/openeo-spring-driver/tree/release/1.2.0"
    },
    {
      "name": "openeo_odc_driver",
      "version": "0.0.1",
      "url": "https://github.com/Open-EO/openeo_odc_driver/tree/ogc_t20"
    }
  ]
}
```

Listing B.1

- `processing:expression` to record the specific mathematical or computational expressions applied during data processing. In this case, the `openeo` format was used including the openEO JSON process graph as processing workflow. Example:

```
{
  "processing:expression": {
    "format": "openeo",
    "expression": {
      "process_graph": {
```

```
        "load1": {
          "process_id": "load_collection",
          "arguments": {
            "id": "ERA5_REANALYSIS",
            "spatial_extent": {
              "west": 13.436072765874142,
              "east": 16.445031081842043,
              "south": 45.399465844609544,
              "north": 46.98533660619179
            },
            "temporal_extent": [
              "2020-01-01T00:00:00Z",
              "2020-12-31T00:00:00Z"
            ],
            "bands": [
              "air_temperature_at_2_metres"
            ],
            "properties": {}
          }
        },
        "save2": {
          "process_id": "save_result",
          "arguments": {
            "data": {
              "from_node": "load1"
            },
            "format": "NETCDF"
          },
          "result": true
        }
      },
      "parameters": []
    }
  }
}
```

Listing B.2

- `processing:facility` to represent the facility that produced the data. Example:

```
{
  "processing:facility": "Eurac Research openEO backend"
}
```

Listing B.3

- `processing:datetime` to log the date and time of each processing task. Example:

```
{
  "processing:datetime": "2024-11-28T11:25:45.931339+00:00"
}
```

Listing B.4

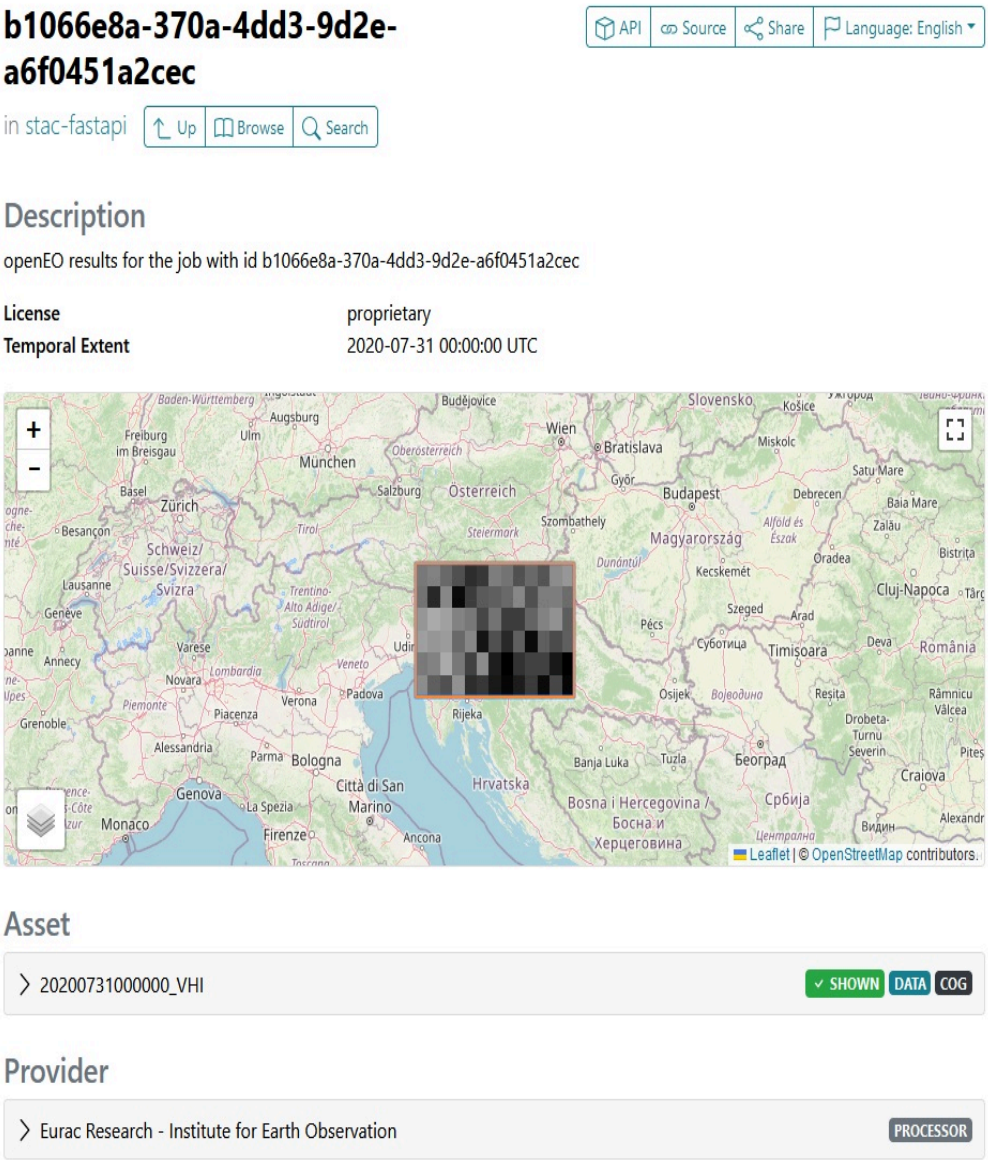

Figure B.1 — STAC Browser showing the openEO batch job result for the VHI workflow

B.2. Observations and Recommendations

The capabilities of the STAC Processing extension make provenance tracking comprehensive and systematic, ensuring that all relevant metadata is preserved alongside the resulting geospatial assets. More fields of the extension could be integrated to improve even further the traceability of the workflow, but due to the limited amount of time the participants were not able to add these fields yet. For instance, it would be advisable to add one or more Links with `derived_from` relationships to the eventual source metadata & data used in the processing. They could be used to trace back the processing history of the dataset.

C

ANNEX C (INFORMATIVE) CRIM GDC PROVENANCE DEMO

ANNEX C (INFORMATIVE) CRIM GDC PROVENANCE DEMO

C.1. Implementation of OGC APIs for GDC Provenance in CRIM Server

For the D144 Provenance Demo deliverable of the *OGC Testbed 20 – GeoDataCubes* task, CRIM provided updates to its Hirondelle development server. These developments included implementing the W3C PROV metadata profiling information into the CRIM *OGC API – Processes* service (https://hirondelle.crim.ca/weaver/) implemented using the Weaver open-source code. See the Weaver – Job Provenance documentation for further details about supported operations and formats.

The PROV metadata is applied through the cwltool library that implements the W3C-PROV standard, and which is the underlying *Common Workflow Language* (CWL) engine employed by CRIM's server to invoke Application Packages from requests to an *OGC API – Processes* execution endpoint. In alignment with the *D140 GeoDataCube API Profile* effort that introduced the `POST /jobs` operation to unify job execution across implementations (openEO, OGC API, etc.), the Provenance operations are integrated under child resources of the corresponding `/jobs` endpoint, such that GDC process execution works seamlessly with Provenance metadata.

In this effort, the draft *OGC API – Processes – Part 4: Job Management* standard was created by a GeoLabs and CRIM collaboration, followed by multiple peer reviews. Improved features were also submitted to the maintainers of cwltool to facilitate the integration of server metadata related to the *OGC API – Processes* implementation running it, such that more adequate and specific provenance information of a job can be tracked across distinct server instances.

C.2. Demonstration of GDC Provenance Operations

To present a succinct and comprehensive demonstration of PROV metadata within *OGC API – Processes* job executions, a simple `echo` process is employed below. This process takes an input string and returns it unmodified. The same concepts would apply to any GDC process invocation. Also, considering that the cwltool library is employed by CRIM's server, any

advanced CWL workflow definition would also be supported to provide PROV metadata for any nested operation involved.

To analyze provenance, the process job must first be executed. This is submitted on `POST /jobs` as follows.

```
{
  "process": "https://hirondelle.crim.ca/weaver/processes/echo",
  "inputs": {
     "message": "Hello World!"
  }
}
```

Listing C.1 — Job Execution

From this execution, a job `Location` header is obtained. This location header can be used as base URL for any of the following Job Provenance requests.

The first endpoint of interest is `GET /jobs/{jobID}/prov`, which contains the PROV-encoded metadata as a Research Object (RO) bundle. This corresponds to the main PROV Data Model entity that is understood by all RO and PROV utilities for sharing metadata about the provenance of data and all underlying agents, entities, activities and attributes it was associated with. Because the PROV definition is wrapped around the job definition in an implementation of *OGC API — Processes*, and that all GDC data processing executions leading to result collections are obtained through a job, this allows providing full traceability of the operations involved in the GDC process.

The `GET /jobs/{jobID}/prov` operation in *Part 4: Job Management* requires only minimal support of PROV-JSON. However, PROV can be requested in multiple formats (PROV-JSON, PROV-JSONLD, PROV-XML, PROV-N, PROV-NT, PROV-TTL, etc.) using the corresponding IANA media-type specified in the `Accept` header.

Because PROV metadata can be very verbose, only the most concise PROV-N representation is shown below. Similar results are obtained with equivalent references and metadata details with the other presentations, although they are not necessarily trivial to convert between them given that other representations (e.g.: XML, JSON-LD) can take advantage of their native schema/context referencing system to link the data differently.

```
document
  prefix wfprov <http://purl.org/wf4ever/wfprov#>
  prefix wfdesc <http://purl.org/wf4ever/wfdesc#>
  prefix cwlprov <https://w3id.org/cwl/prov#>
  prefix foaf <http://xmlns.com/foaf/0.1/>
  prefix schema <http://schema.org/>
  prefix orcid <https://orcid.org/>
  prefix id <urn:uuid:>
  prefix data <urn:hash::sha1:>
  prefix sha256 <nih:sha-256;>
  prefix researchobject <arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/>
  prefix metadata <arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/metadata/>
  prefix provenance <arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/metadata/
provenance/>
  prefix wf <arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/workflow/packed.
cwl#>
```

```
  prefix input <arcp://uuid,6617474d-4dc1-4f77-a993-2527c2618a7c/workflow/
primary-job.json#>
  prefix doi <https://doi.org/>
  prefix wf4ever <http://purl.org/wf4ever/wf4ever#>

  agent(id:748357d2-0b70-454a-a342-c960c25fb4c6)
  agent(id:748357d2-0b70-454a-a342-c960c25fb4c6, [prov:type='foaf:
OnlineAccount', prov:location="https://hirondelle.crim.ca/weaver", cwlprov:
hostname="hirondelle.crim.ca"])
  agent(id:748357d2-0b70-454a-a342-c960c25fb4c6, [prov:type='foaf:
OnlineAccount', prov:label="crim-ca/weaver:6.1.1", foaf:accountName="crim-ca/
weaver:6.1.1"])
  agent(id:89a542bc-fdf0-4cb6-ba73-cef27225c117, [prov:type='schema:Person',
prov:type='prov:Person', prov:label="crim-ca/weaver:6.1.1", foaf:name="crim-ca/
weaver:6.1.1", foaf:account='id:748357d2-0b70-454a-a342-c960c25fb4c6', schema:
name="crim-ca/weaver:6.1.1"])
  actedOnBehalfOf(id:748357d2-0b70-454a-a342-c960c25fb4c6, id:89a542bc-fdf0-4cb6-
ba73-cef27225c117, -)
  agent(id:d8f3560c-c86b-46c0-9dd5-de85de07db1b, [prov:type='prov:
SoftwareAgent', prov:type='wfprov:WorkflowEngine', prov:label="cwltool
3.1.20241217163858"])
  wasStartedBy(id:d8f3560c-c86b-46c0-9dd5-de85de07db1b, -, id:748357d2-0b70-454a-
a342-c960c25fb4c6, 2024-12-21T05:46:04.879734)
  activity(id:6617474d-4dc1-4f77-a993-2527c2618a7c, 2024-12-21T05:46:04.879778, -
, [prov:type='wfprov:WorkflowRun', prov:label="Run of workflow/packed.cwl#main"])
  wasAssociatedWith(id:6617474d-4dc1-4f77-a993-2527c2618a7c, id:d8f3560c-c86b-
46c0-9dd5-de85de07db1b, wf:main)
  wasStartedBy(id:6617474d-4dc1-4f77-a993-2527c2618a7c, -, id:d8f3560c-c86b-46c0-
9dd5-de85de07db1b, 2024-12-21T05:46:04.879859)
  entity(data:644e201526525f62152815a76a2dc773450f3dd9, [prov:type='prov:
PrimarySource', prov:label="Source code repository", prov:location="https://
github.com/crim-ca/weaver"])
  agent(data:4e5feeeb8209de47c8dfb7c3f50a893e505af067, [prov:type='prov:
SoftwareAgent', prov:location="https://hirondelle.crim.ca/weaver", prov:label=
"crim-ca/weaver:6.1.1", prov:label="Weaver is an Execution Management Service
(EMS) that allows the execution of workflows chaining various applications and
Web Processing Services (WPS) inputs and outputs. Remote execution is deferred
by the EMS to an Application Deployment and Execution Service (ADES), as defined
by Common Workflow Language (CWL) configurations.", prov:generalEntity='data
:644e201526525f62152815a76a2dc773450f3dd9', prov:specificEntity='doi:10.5281/
zenodo.14210717'])
  entity(data:3102f6d7a018ebae572f457d711ed7e1e7a11bc2, [prov:type='prov:
Organization', foaf:name="Computer Research Institute of Montréal", schema:name=
"Computer Research Institute of Montréal"])
  entity(data:838cdfa4bbf09d1aedd26d79b46bfa8778ede2e0, [foaf:name="crim-ca/
weaver", schema:name="crim-ca/weaver", prov:location="http://pavics-weaver.
readthedocs.org/en/latest/", prov:type='prov:Organization', prov:label="Server
Provider"])
  entity(id:6617474d-4dc1-4f77-a993-2527c2618a7c, [prov:type='wfdesc:
ProcessRun', prov:location="https://hirondelle.crim.ca/weaver/processes/echo/
jobs/6617474d-4dc1-4f77-a993-2527c2618a7c", prov:label="Job Information"])
  entity(data:4e5feeeb8209de47c8dfb7c3f50a893e505af067:echo, [prov:type='wfdesc:
Process', prov:location="https://hirondelle.crim.ca/weaver/processes/echo", prov:
label="Process Description"])
  wasDerivedFrom(data:4e5feeeb8209de47c8dfb7c3f50a893e505af067, data:644e20152652
5f62152815a76a2dc773450f3dd9, -, -, -, [prov:type='prov:PrimarySource'])
  actedOnBehalfOf(data:4e5feeeb8209de47c8dfb7c3f50a893e505af067, id:89a542bc-
fdf0-4cb6-ba73-cef27225c117, -)
  specializationOf(data:4e5feeeb8209de47c8dfb7c3f50a893e505af067, id:748357d2-
0b70-454a-a342-c960c25fb4c6)
  wasAttributedTo(data:3102f6d7a018ebae572f457d711ed7e1e7a11bc2, data:644e2015265
25f62152815a76a2dc773450f3dd9)
```

```
  wasDerivedFrom(id:748357d2-0b70-454a-a342-c960c25fb4c6, data:4e5feeeb8209de47c8
dfb7c3f50a893e505af067, -, -, -)
  wasStartedBy(id:6617474d-4dc1-4f77-a993-2527c2618a7c, data:4e5feeeb8209de47c8df
b7c3f50a893e505af067, -, -)
  wasStartedBy(id:d8f3560c-c86b-46c0-9dd5-de85de07db1b, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, -, 2024-12-21T05:46:02.784000+00:00)
  specializationOf(id:d8f3560c-c86b-46c0-9dd5-de85de07db1b, id:6617474d-4dc1-
4f77-a993-2527c2618a7c)
  alternateOf(id:d8f3560c-c86b-46c0-9dd5-de85de07db1b, id:6617474d-4dc1-4f77-
a993-2527c2618a7c)
  wasGeneratedBy(id:6617474d-4dc1-4f77-a993-2527c2618a7c, data:4e5feeeb8209de47c8
dfb7c3f50a893e505af067:echo, -)
  wasDerivedFrom(data:838cdfa4bbf09d1aedd26d79b46bfa8778ede2e0, data:4e5feeeb8209
de47c8dfb7c3f50a893e505af067, -, -, -)
  wasAttributedTo(data:838cdfa4bbf09d1aedd26d79b46bfa8778ede2e0, data:4e5feeeb820
9de47c8dfb7c3f50a893e505af067)
  entity(wf:main, [prov:type='prov:Plan', prov:type='wfdesc:Process', prov:label=
"Prospective provenance"])
  entity(data:2ef7bde608ce5404e97d5f042f95f89f1c232871, [prov:type='wfprov:
Artifact', prov:value="Hello World!"])
  used(id:6617474d-4dc1-4f77-a993-2527c2618a7c, data:2ef7bde608ce5404e97d5f042f95
f89f1c232871, 2024-12-21T05:46:04.883022, [prov:role='wf:main/message'])
  agent(id:4a05bebf-124c-4cf2-ada1-4d3bc6ecc3d4, [prov:type='prov:
SoftwareAgent', cwlprov:image="debian:stretch-slim", prov:label="Container
execution of image debian:stretch-slim"])
  wasAssociatedWith(id:6617474d-4dc1-4f77-a993-2527c2618a7c, id:4a05bebf-124c-
4cf2-ada1-4d3bc6ecc3d4, -)
  entity(data:2ef7bde608ce5404e97d5f042f95f89f1c232871, [prov:type='wfprov:
Artifact', prov:value="Hello World!"])
  used(id:6617474d-4dc1-4f77-a993-2527c2618a7c, data:2ef7bde608ce5404e97d5f042f95
f89f1c232871, 2024-12-21T05:46:04.940812, [prov:role='wf:main/echo/message'])
  entity(data:a0b65939670bc2c010f4d5d6a0b3e4e4590fb92b, [prov:type='wfprov:
Artifact'])
  entity(id:c2ffa0b6-271a-4a07-8070-9ba6db93893e, [prov:type='wf4ever:File',
prov:type='wfprov:Artifact', cwlprov:basename="stdout.log", cwlprov:nameroot=
"stdout", cwlprov:nameext=".log"])
  specializationOf(id:c2ffa0b6-271a-4a07-8070-9ba6db93893e, data:a0b65939670bc2c0
10f4d5d6a0b3e4e4590fb92b)
  wasGeneratedBy(id:c2ffa0b6-271a-4a07-8070-9ba6db93893e, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.207918, [prov:role='wf:main/echo/output'])
  entity(data:da39a3ee5e6b4b0d3255bfef95601890afd80709, [prov:type='wfprov:
Artifact'])
  entity(id:16789964-9acd-4a91-b636-5ee7eca20011, [prov:type='wf4ever:File',
prov:type='wfprov:Artifact', cwlprov:basename="stderr.log", cwlprov:nameroot=
"stderr", cwlprov:nameext=".log"])
  specializationOf(id:16789964-9acd-4a91-b636-5ee7eca20011, data:da39a3ee5e6b4b0d
3255bfef95601890afd80709)
  wasGeneratedBy(id:16789964-9acd-4a91-b636-5ee7eca20011, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.207918, [prov:role='wf:main/echo/PACKAGE_
OUTPUT_HOOK_LOG_b563c5f1-d4e0-46f0-b5ea-6b20b8aff93a'])
  entity(data:a0b65939670bc2c010f4d5d6a0b3e4e4590fb92b)
  entity(id:6cad4e06-b51d-442d-b480-9e0199513ef7, [prov:type='wf4ever:File',
prov:type='wfprov:Artifact', cwlprov:basename="stdout.log", cwlprov:nameroot=
"stdout", cwlprov:nameext=".log"])
  specializationOf(id:6cad4e06-b51d-442d-b480-9e0199513ef7, data:a0b65939670bc2c0
10f4d5d6a0b3e4e4590fb92b)
  wasGeneratedBy(id:6cad4e06-b51d-442d-b480-9e0199513ef7, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.207918, [prov:role='wf:main/echo/stdout.
log'])
  wasEndedBy(id:6617474d-4dc1-4f77-a993-2527c2618a7c, -, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.207897)
```

```
  wasGeneratedBy(id:c2ffa0b6-271a-4a07-8070-9ba6db93893e, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.211805, [prov:role='wf:main/primary/
output'])
  wasGeneratedBy(id:16789964-9acd-4a91-b636-5ee7eca20011, id:6617474d-4dc1-
4f77-a993-2527c2618a7c, 2024-12-21T05:46:06.211805, [prov:role='wf:main/primary/
PACKAGE_OUTPUT_HOOK_LOG_b563c5f1-d4e0-46f0-b5ea-6b20b8aff93a'])
  wasGeneratedBy(id:6cad4e06-b51d-442d-b480-9e0199513ef7, id:6617474d-4dc1-4f77-
a993-2527c2618a7c, 2024-12-21T05:46:06.211805, [prov:role='wf:main/primary/
stdout.log'])
  wasEndedBy(id:6617474d-4dc1-4f77-a993-2527c2618a7c, -, id:d8f3560c-c86b-46c0-
9dd5-de85de07db1b, 2024-12-21T05:46:06.212038)
endDocument
```

Listing C.2 — Job Provenance encoded in PROV-N

The other endpoints all correspond to sub-operations provided by cwlprov. At the time of writing, those operations are not yet defined in the *OGC API — Processes Part 4: Job Management* standard, but could easily be added since they would be demonstrated by multiple testbed participants.

C.3. Observations and Recommendations

Although the PROV metadata was successfully implemented in Weaver to demonstrate that implementations of *OGC API — Processes — Part 4: Job Management* can effectively provide provenance information in the context of OGC APIs and GDC, the draft standard still lacks details regarding “how” provenance is effectively applied. For example, there is no explicit requirement for the `jobID` and `runID` of nested workflow executions to align. Similarly, there is no requirement for the `jobID` itself to align with the UUID assigned to the *Research Object*, nor any mandatory metadata related to the server, the instance, the reference implementation, or any other relevant information that would be critical for proper provenance interpretation of GDC results.

Therefore, while PROV establishes the right concepts for more trustworthy and understandable data provenance, the integrity of GDC remains relatively poorly defined. It relies on the assumption that each server implementation will act with good intent and provide sufficient detail to trace the complete data lineage. It is recommended that the OGC develop clearer guidelines for implementing PROV within the context of OGC API Standards, to enhance the interoperability of data provenance.

D

ANNEX D (INFORMATIVE) GEOLABS GDC PROVENANCE DEMO

ANNEX D (INFORMATIVE) GEOLABS GDC PROVENANCE DEMO

GeoLabs contributed to the OGC Testbed 20 GeoDataCube D144 Provenance Demonstration task by initiating discussions on aligning job management between an OGC API—Processes endpoint and an openEO server implementation, with a focus on enabling provenance tracking. These discussions led to the initial draft of the OGC API—Processes—Part 4: Job Management specification.

Building on work completed during Testbed 19 last year, a prototype server instance combining the ZOO-Project-DRU (Deploy, Replace, and Undeploy) and eoAPI was used as a foundation to implement new capabilities. These included managing jobs through the additional endpoints defined in the Part 4 draft and supporting provenance tracking. eoAPI is an open-source, reusable framework designed to harness Earth observation data.

D.1. OGC API — Processes — Part 4: Job Management

The OGC API — Processes — Part 4: Job Management draft standard targeted alignment with openEO batch processing tasks. It extends the OGC API — Processes — Part 1: Core Standard, where the POST `/processes/{processId}/execution` operation creates a `/jobs/{jobId}` corresponding to the ongoing execution of the designated process.

The Part 4 extension defines how to use the `POST /jobs` operation to create jobs waiting for execution, then the `POST /jobs/{jobId}` operation to trigger the effective execution of the task.

When creating a job, the `Content-Type` is set to `application/json` and the `Content-Schema` can be used to determine whether the job is an openEO process graph or an OGC API — Processes — Part 3: Workflow execute request with the `process` key. The `Profile` HTTP header was also discussed and investigated during the activity to help servers determine the job's execution profile.

The last endpoint does not require any body content, and the jobs endpoints, defined in the OGC API — Processes — Part 1: Core Standard, permit access to the jobs' execution status and results. However, a JSON object with a provenance parameter set to true can be associated with the request to activate the provenance registration.

A new `GET /jobs/{jobId}/prov` operation was added to access the provenance information encoded and formatted per W3C PROV-JSON for jobs started with the provenance tracking

activated. Content negotiation permits the support of other formats: PROV-O as JSON-LD, PROV-XML, and PROV-N.

D.2. Prototype Server Instance

A prototype Server Instance is available at https://tb20.geolabs.fr:8001/ogc-api/. Combining eoAPI and ZOO-Project-DRU APIs to expose a single API providing Data Access as STAC (from stac-fastapi) and processing capabilities as OGC API — Processes.

The prototype Server Instance supports the following OGC Standards:

- OGC API — Processes — Part 1: Core Standard (Reference Implementation)
- OGC API — Processes — Part 2: Deploy, Replace, Undeploy; Part 3: Workflow (remote-processes conformance class); and Part 4: Job Management draft specification
- OGC API — Features — Part 1: Core and Part 3: Filtering Standards

Geolabs used a Keycloack server to secure access to some of the endpoints mentioned before. Geolabs also implemented the /credential/oidc and /me endpoints to ensure interoperability with client applications (e.g., gdc-web-editor).

The Geolabs implementation relies on the cwltool to capture the provenance of workflow execution. For the prototype Server Instance to support the provenance conformance class defined in the OGC API — Processes — Part 4: Job Management, it requires invoking the execution of a process deployed using an implementation of the the OGC API — Processes — Part 2: Deploy, Replace, Undeploy draft specification (cwl conformance class).

The STAC catalog associated with the processing engine stores the execution results as a Collection identified by the job identifier. There is no security in place to access the processing results.

The water-bodies CWL OGC Application Package was modified to illustrate how to add the processing STAC extension metadata information to the STAC item resulting from the processing. It adds the following properties:

- `processing:datetime` determining when the processing occurred,
- `processing:facility` the computing resources where the job runs,
- `processing:software` list of software involved in the processing,
- `processing:expression` is an object encapsulating the request used when creating the job through the /jobs endpoint.

Below are the properties associated with a STAC item corresponding to the job id `d28748d0-de49-11ef-b0db-0242ac1f0008`, which was created by posting to the `/jobs` endpoint, as

defined in the OGC API — Processes — Part 4: Job Management, run on the prototype Server Instance.


```
"processing:software": {
    "ZOO-Project-DRU": "2.0.1",
    "cwltool": "3.1.20240508115724",
    "ghcr.io/terradue/ogc-eo-application-package-hands-on/stage": "1.3.2",
    "ghcr.io/terradue/ogc-eo-application-package-hands-on/stac": "1.5.0",
    "ghcr.io/terradue/ogc-eo-application-package-hands-on/crop": "1.5.0",
    "ghcr.io/terradue/ogc-eo-application-package-hands-on/norm_diff": "1.5.0",
    "ghcr.io/terradue/ogc-eo-application-package-hands-on/otsu": "1.5.0"
},
"processing:expression": {
    "format": "ogc-api-processes-3",
    "expression": {
        "process": "https://tb20.geolabs.fr:8001/ogc-api/processes/water-bodies",
        "inputs": {
            "stac_items": [
                "https://earth-search.aws.element84.com/v0/collections/sentinel-
s2-l2a-cogs/items/S2B_10TFK_20210713_0_L2A",
                "https://earth-search.aws.element84.com/v0/collections/sentinel-
s2-l2a-cogs/items/S2A_10TFK_20220524_0_L2A"
            ],
            "aoi": "-121.399,39.834,-120.74,40.472",
            "epsg": "EPSG:4326",
            "bands": [
                "green",
                "nir"
            ]
        }
    }
}
"processing:facility": "GeoLabs - D144 - Provenance demonstration - OGC Testbed-
20 dedicated ressources",
"processing:datetime": "2025-01-29T14:06:55.198Z",
```


Listing D.1 — STAC Item processing metadata

Adding this metadata information to the STAC item from within the CWL OGC Application Package is not a solution as it implies adapting all the CWL before deployment. Using the provenance information stored during the execution to publish the STAC items with relevant processing metadata makes more sense. In the following shows where such information is accessible from the provenance generated by cwltool.

Starting from the `/jobs/{jobId}/prov` operation, accessible only for authenticated user the main workflow provenance encoded in PROV-JSON format is accessed. In the `agent` object extracted below, the `prov:location` and `cwlprov:hostname` fields that provide information about the execution environment and the hostname can be seen. The example also shows that the `prov:label` and `foaf:accountName` fields identify the user who executed the workflow on the host. Then there is one `wfprov:WorkflowEngine` and `prov:SoftwareAgent` with `prov:label` field that identifies the software agent used to execute the workflow. Finally, the `cwlprov:image` field of the last `agent` provides information about the Docker image version used to execute this part of the workflow.


```
"agent": {
    "id:b4118585-ef58-4a13-99cd-2a89fba04e10": [
        {},
        {
        "prov:type": {
```

```
            "$": "foaf:OnlineAccount",
            "type": "prov:QUALIFIED_NAME"
        },
        "prov:location": "crdp.geolabs.fr",
        "cwlprov:hostname": "crdp.geolabs.fr"
        },
        {
        "prov:type": {
            "$": "foaf:OnlineAccount",
            "type": "prov:QUALIFIED_NAME"
        },
        "prov:label": "gerald.fenoy",
        "foaf:accountName": "gerald.fenoy"
        }
    ],
    "id:60ba1ff6-afc1-4733-9659-07a6e85d28fb": {
        "prov:type": {
            "$": "prov:Person",
            "type": "prov:QUALIFIED_NAME"
        },
        "prov:label": "",
        "foaf:name": "",
        "foaf:account": {
            "$": "id:b4118585-ef58-4a13-99cd-2a89fba04e10",
            "type": "prov:QUALIFIED_NAME"
        }
    },
    "id:c7134721-0c1d-4904-ac15-44e440e2f08c": {
        "prov:type": [
        {
            "$": "wfprov:WorkflowEngine",
            "type": "prov:QUALIFIED_NAME"
        },
        {
            "$": "prov:SoftwareAgent",
            "type": "prov:QUALIFIED_NAME"
        }
        ],
        "prov:label": "cwltool 3.1.20240508115724"
    },
    "id:f4d53b7a-cb4d-48de-ae03-b41510e88bce": {
        "prov:type": {
            "$": "prov:SoftwareAgent",
            "type": "prov:QUALIFIED_NAME"
        },
        "cwlprov:image": "ghcr.io/terradue/ogc-eo-application-package-hands-on/
stage:1.3.2",
        "prov:label": "Container execution of image ghcr.io/terradue/ogc-eo-
application-package-hands-on/stage:1.3.2"
    }
},
```


Listing D.2 — agent property from main workflow provenance

As shown below, in the `wfprov:ProcessRun` from the `activity` of the previous provenance information there is a link to another provenance file.


```
"id:80180aba-716c-44fd-bdb7-6f454c41862d": [
    {
        "prov:type": {
            "$": "wfprov:ProcessRun",
```

```
            "type": "prov:QUALIFIED_NAME"
        },
        "prov:label": "Run of workflow/packed.cwl#main/main/on_stage"
    },
    {
    "prov:has_provenance": [
        [...]
        {
            "$": "provenance:workflow_20on_stage.80180aba-716c-44fd-bdb7-
6f454c41862d.cwlprov.json",
            "type": "prov:QUALIFIED_NAME"
        },
        [...]
    ]
    }
],
```

Listing D.3 — `wfprov:ProcessRun activity` from the main workflow provenance

The file contains the following additional element in the `agent` object.

```
"id:eed7a749-e94b-4968-8059-f756e07f92eb": {
    "prov:type": {
        "$": "prov:SoftwareAgent",
        "type": "prov:QUALIFIED_NAME"
    },
    "cwlprov:image": "ghcr.io/terradue/ogc-eo-application-package-hands-on/
stac:1.5.0",
    "prov:label": "Container execution of image ghcr.io/terradue/ogc-eo-
application-package-hands-on/stac:1.5.0"
}
```

Listing D.4 — additional property from the `agent` object in the workflow step provenance

In the `wfprov:ProcessRun` from the activity of the previous provenance information there are two links to other provenance files: sample1 and sample2. One contains the following additional information.

```
"id:5df11816-1002-4e16-8f58-40bfab5b5133": {
    "prov:type": {
        "$": "prov:SoftwareAgent",
        "type": "prov:QUALIFIED_NAME"
    },
    "cwlprov:image": "ghcr.io/terradue/ogc-eo-application-package-hands-on/
crop:1.5.0",
    "prov:label": "Container execution of image ghcr.io/terradue/ogc-eo-
application-package-hands-on/crop:1.5.0"
},
"id:ba1d74b9-e43c-4213-baaf-628d8f98bdd4": {
    "prov:type": {
        "$": "prov:SoftwareAgent",
        "type": "prov:QUALIFIED_NAME"
    },
    "cwlprov:image": "ghcr.io/terradue/ogc-eo-application-package-hands-on/norm_
diff:1.5.0",
    "prov:label": "Container execution of image ghcr.io/terradue/ogc-eo-
application-package-hands-on/norm_diff:1.5.0"
},
"id:46155cb4-7c9c-491b-ae05-09085be32d07": {
    "prov:type": {
```

```
        "$": "prov:SoftwareAgent",
        "type": "prov:QUALIFIED_NAME"
    },
    "cwlprov:image": "ghcr.io/terradue/ogc-eo-application-package-hands-on/
otsu:1.5.0",
    "prov:label": "Container execution of image ghcr.io/terradue/ogc-eo-
application-package-hands-on/otsu:1.5.0"
}
```

Listing D.5 — additional property from the workflow step provenance

In the last provenance files, there is no more `wfprov:ProcessRun` with `prov:has_provenance` ans as such the provenance metadata stop here.

Using the `cwlprov` python package, detailed information can be accessed using the provenance directory produced by the `cwltool` at runtime. The following commands are available: `validate`, `info`, `who`, `inputs`, `outputs`, `run`, `runs`.

On the Server Instance environment, `cwlprov` cannot access the produced directory due to multiple definitions with conflicting values on some manifest files. If the manifests are manually corrected to pass the `validate` command, the result of the `run` command does not give the same result on every run depending on the `{runId}` used.

Due to these technical challenges, further implementation or investigation of the other operations discussed with other participants during this Testbed 20 activity was not pursued.

D.3. Observations and Recommendations

As illustrated in this activity, starting from an OGC Application Package, it is possible to combine;

- Implementations of the OGC API — Processes Standard,
- To deploy using implementations of the OGC API — Processes — Part 2 draft specification, and
- Execute processes through implementations of the OGC API — Processes — Part 1 Standard, or Part 4: Job Management, with STAC catalog, to store provenance metadata information about the job responsible for generating the STAC Items from the STAC collection resulting from process execution.

Geolabs demonstrated that it is possible to extract the attributes defined in the processing stac extension from the provenance metadata files generated by `cwltool` to set them in the STAC items in the STAC.

The STAC processing extension presents some limitations, especially regarding the remote-processes conformance class. In such a case, the provenance metadata may need to provide information about multiple `processing:facility` and versions of the `processing:software` used, corresponding to the processing engine. Consider a trivial process chain involving the

CRIM and GeoLabs Server Instances. It is unclear how the steps executed on the CRIM server should be reported from the GeoLabs provenance information and vice versa.

The OGC API — Processes — Part 4: Job Management draft specification should describe the remote process chain's provenance tracking as defined in the OGC API — Processes — Part 3: Workflow draft specification when relying on remote processes.

Following the OGC Best Practic for Earth Observation Application Package, if the OGC API — Processes processDescription of a deployed process contains only a single output, it can be converted into an openEO process. More experimentation should be done to determine to what extent an OGC Application Package can be described as an openEO process, that would permit chaining with other openEO processes. This would be a significant step towards interoperability between implementations of the OGC API — Processes and openEO standards. This would require that openEO permit defining processes graphs involving multiple openEO Server Instances.

The STAC Items resulting from process execution and publication would benefit from more discussions for its introduction as a recommendation in the context of the OGC API — Processes — Part 4: Job Management draft specification. If the provenance tracking is activated when starting the job, the STAC Items may be available with the provenance information associated with the job that generated them.

When reproducibility matters, the scientific citation STAC extension can be used based on the metadata information available from the deployed CWL OGC Application Package.