OGC® DOCUMENT: 24-051

External identifier of this OGC® document: http://www.opengis.net/doc/IS/ogcapi-processes-4/1.0

OGC API - PROCESSES - PART 4: JOB MANAGEMENT

STANDARD
Implementation

DRAFT

Version: 1.0
Submission Date: yyyy-mm-dd
Approval Date: yyyy-mm-dd
Publication Date: yyyy-mm-dd
Editor: Gérald Fenoy

CONTENTS

I ABSTRACT

OGC API Standards define modular API building blocks to spatially enable Web APIs in a consistent way. The OpenAPI specification is used to define the API building blocks.

The OGC API Processes Standard (aka Processes API) defines API building blocks to describe, execute, monitor and retrieve results of Web-accessible processes. OGC API Processes is comprised of multiple parts, each of them is a separate OGC Standard.

The OGC API — Processes — Part 2: Deploy, Replace, Undeploy draft specification extends the core capabilities specified in OGC API — Processes — Part 1: Core [OGC 18-062r2] with the ability to dynamically add, modify and/or delete individual processes using an implementation (endpoint) of the OGC API — Processes Standard.

The OGC API — Processes — Part 3: Workflows draft specification extends the core capabilities specified in OGC API — Processes — Part 1: Core [OGC 18-062r2] with the ability to …

The OGC API — Processes — Part 4: Job Management draft specification extends the core capabilities specified in OGC API — Processes — Part 1: Core [OGC 18-062r2] with the ability to create, manage and monitor jobs that are associated with processes execution. This part of the standard also defines how to ensure provenance information is preserved and findable.

CAUTION

This is a DRAFT version of the 4th part of the OGC API — Processes standards. This draft is not complete and there are open issues that are still under discussion.

II KEYWORDS

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

process, instance, spatial, data, openapi, job, create, update, delete, add, remove, REST, PATCH, POST, DELETE

III SECURITY CONSIDERATIONS

See OGC API — Processes — Part 1: Core, Clause 10.4.

Since creating and updating jobs will change the jobs available to a client, servers will — in almost all cases — restrict the access to these operations.

Users making modifications to job resources need to:

1. Be authenticated,
2. Have “modification privileges” on the jobs offered through the API,
3. Have access to one or more of the POST and/or PATCH methods on the jobs `/job` and `/jobs/{jobID}` endpoints.

The API definition, as defined in Clause 7.3 from OGC 18-062r2, must reflect this in the resource paths and their available methods.

IV SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Geolabs
- Terradue Srl.
- Centre de Recherche Informatique de Montréal (CRIM)

V SUBMITTERS

All questions regarding this submission should be directed to the editors or the submitters:

NAME	AFFILIATION
Gérald Fenoy *(editor)*	GeoLabs
Francis Charette-Migneault *(editor)*	Centre de Recherche Informatique de Montréal (CRIM)
Panagiotis (Peter) A. Vretanos	CubeWerx Inc.

1

SCOPE

1 SCOPE

The OGC API — Processes — Part 4 Standard is an extension to the OGC API – Processes – Part 1: Core Standard [OGC 18-062r2] and defines the behavior of a server that supports the ability to create jobs without implying the process execution starts right away.

Specifically, the Processes Part 4 Standard specifies:

- How to manage job.
- How to handle provenenance information associated with a job.

2

CONFORMANCE

2 CONFORMANCE

The OGC API — Processes — Part 4 Standard defines the following requirements classes.

The main requirements class is:

- Job Management.

This class specifies requirements that any Web API implementing Processes Part 4 must conform with.

The `Job Management` class does not mandate a specific encoding for the job definition. However, the Part 4 extension defines the following conformance class:

- OGC API — Processes — Workflow Execute Request
- OpenEO Process Graph

The `OGC API - Processes - Workflow Execute Request` class defines that jobs can be created from an OGC API — Processes — Workflow Execute Request.

The `OpenEO Process Graph` class defines that jobs can be created from an OpenEO Process Graph.

The provenance information associated with a job is not mandated to be supported by the server. A dedicated requirements class Provenance is defined for this feature.

The standardization target for all Conformance class defined in this Standard is “Web API”.

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Benjamin Pross, Panagiotis (Peter) A. Vretanos: OGC 18-062r2, *OGC API — Processes — Part 1: Core*. Open Geospatial Consortium (2021). http://www.opengis.net/doc/IS/ogcapi-processes-1/1.0.0.

Fenoy, G.: OGC 20-044, **OGC API — Processes — Part 2: Deploy, Replace, Undeploy**

Jacovella-St-Louis J.: OGC 21-009, **OGC API — Processes — Part 3: Workflows**

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r3, *OGC API — Features — Part 1: Core*. Open Geospatial Consortium (2019). http://www.opengis.net/doc/IS/ogcapi-features-1/1.0.0.

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

4 TERMS, DEFINITIONS AND ABBREVIATED TERMS

No terms and definitions are listed in this document.

4.1. Terms and definitions

See OGC 18-062r2, Clause 4.1.

4.2. Abbreviated terms

See OGC 18-062r2, Clause 4.2.

5 CONVENTIONS AND BACKGROUND

5 CONVENTIONS AND BACKGROUND

See OGC 18-062r2, Clause 5.

6

REQUIREMENTS CLASS “JOB MANAGEMENT”

6 REQUIREMENTS CLASS “JOB MANAGEMENT”

6.1. Overview

REQUIREMENTS CLASS 1	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITES	OGC API — Processes — Part 1: Core, Conformance Class ‘core’ RFC 2616 (HTTP/1.1)
LABEL	http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/job-management

A server that implements the Job Management Requirements Class provides the ability to create, update and start jobs.

The HTTP POST method on the `/jobs` path is used on to create new jobs.

The HTTP PATCH method on the `/jobs/{jobID}` is used to update the definition of a previously created jobs that are accessible via the Processes API endpoint.

Finally, the HTTP POST method on the `/job/{jobID}/results` is used to start a job.

Creating or updating a job requires that a formal description of the new or updated jobs be provided by the client. This Standard does not mandate that a specific jobs schema be used. However, this extension defines the following conformance classe:

- OGC API — Processes — Wrokflow Execute Request, that enables support for OGC API — Processes — Part 3: Wofklows for jobs definitions.
- OpenEO Process Graph, that enables support for OpenEO-encoded jobs definitions.

6.1.1. HTTP status codes

Clients implementing the Processes API Part 4 should be prepared to handle any legal HTTP or HTTPS status code.

The **Status Codes** listed in Table 1 are of particular relevance to implementors of this Standard. Status codes 200, 201 and 404 are called out in API requirements. Therefore, support for these status codes is mandatory for all compliant implementations. The remainder of the status codes in Table 1 are not mandatory, but are important for the implementation of a well functioning API implementation. Support for these status codes is strongly encouraged for both client and server implementations.

Table 1 — Typical HTTP status codes

STATUS CODE	DESCRIPTION
`200`	A successful request.
`201`	The server has successfully completed the operation and a new resource has been created.
`202`	The request was accepted for processing, but the processing was not completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.
`204`	A successful request with no additional content to send in the response.
`400`	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
`401`	The request requires user authentication. The response includes a `WWW-Authenticate` header field containing a challenge applicable to the requested resource.
`403`	The server understood the request, but is refusing to execute the request. While status code `401` indicates missing or bad authentication, status code `403` indicates that authentication is not the issue, but that the client is not authorized to perform the requested operation on the resource.
`404`	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
`405`	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
`406`	Content negotiation failed. For example, the `Accept` header submitted in the request did not support any of the media types supported by the server for the requested resource.
`412`	The status code indicates that one or more conditions given in the request header fields evaluated to false when tested by the server.
`413`	The server is refusing to process the request because the request content is larger than the server is willing or able to process.
`415`	The server is refusing to service the request because the content is in a format not supported by this method on the target resource.
`422`	The server understands the content type of the request content and the syntax of the request content is correct, but was unable to process the contained instructions. For example, the submitted resource does not meet a semantic constraint, e.g. a mandatory property is missing.

STATUS CODE	DESCRIPTION
`500`	An internal error occurred in the server.

NOTE: Status code `422` is not yet an official HTTP status code, but is expected to be added by the draft IETF RFC “HTTP Semantics”.

More specific guidance is provided for each resource, where applicable.

PERMISSION 1	
LABEL	/per/job-management/additional-status-codes
A	Servers MAY support other HTTP protocol capabilities. Therefore, the server may return other status codes than those listed in Table 1.

The API Description Document describes the HTTP status codes generated by that API imeplementation instance. This should not be an exhaustive list of all possible status codes. It is not reasonable to expect an API designer to control the use of HTTP status codes which are not generated by their software. Therefore, it is recommended that the API Description Document be limited to describing HTTP status codes relevant to the proper operation of the API application logic. Client implementations should be prepared to receive HTTP status codes in addition to those described in the API Description Document.

6.1.2. Cross-origin requests

See OGC API — Features — Part 1: Core, section Support for cross-origin requests, about the importance of supporting cross-origin requests, typically via Cross-origin resource sharing (CORS).

6.2. Creating a new job

6.2.1. Sequence diagram

The following diagram illustrates the sequence diagram for deploying a new process to the API:

```
Client                                                          Server
  |                                                               |
  |   POST /jobs   HTTP/1.1                                       |
  |   Content-Type: application/json                              |
  |                                                               |
  |------------------------------------------------------------->|
  |                                                               |
  |   HTTP/1.1 201 Created                                             |
  |   Location: /jobs/{jobID}                                          |
  |<-------------------------------------------------------------      |
```

Listing 1

6.2.2. Operation

REQUIREMENT 1	
LABEL	/req/job-management/create-post-op
A	The server SHALL support the HTTP POST operation at the path `/jobs`.

6.2.3. Request body

6.2.3.1. Overview

REQUIREMENT 2	
LABEL	/req/job-management/create-body
A	The body of the POST request SHALL be in JSON format.

PERMISSION 2	
LABEL	/per/job-management/create-body
A	A server MAY support any encoding in the body of a HTTP POST operation.

REQUIREMENT 3	
LABEL	/req/job-management/create-content-type
A	The `Content-Type` header SHALL be used to declare the media type of the request.

See section 3.1.1.5 of RFC 7231 for details.

PERMISSION 3	
LABEL	/per/job-management/create-content-schema
A	The `Content-Schema` header MAY be an URI to a JSON or OpenAPI 3.0 Schema document that describes the structure of the request body.

6.2.3.2. OGC API — Processes — Workflow Execute Request body

RECOMMENDATION 1	
LABEL	/rec/job-management/create-body-ogcapi-processes
A	If the job can be encoded as an OGC API — Processes — Workflow Execute Request, implementation SHOULD consider supporting the OGC API — Processes — Workflow Execute Request encoding.

6.2.3.3. OpenEO Process Graph body

RECOMMENDATION 2	
LABEL	/rec/job-management/create-body-openeo
A	If the job can be encoded as an OpenEO graph, implementation SHOULD consider supporting the OpenEO graph encoding.

6.2.4. Response

REQUIREMENT 4	
LABEL	/req/job-management/create-response-jobid
A	If the operation completes, the server SHALL generate a job identifier (i.e. `{jobID}`) for the created job.

REQUIREMENT 5	
LABEL	/req/job-management/create-response-success

REQUIREMENT 5	
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `201`.
B	A response with HTTP status code `201` SHALL include a `Location` header with the URI of the deployed processes (path: `/jobs/{jobID}`).

REQUIREMENT 6	
LABEL	/req/job-management/create-response-body
A	The response SHALL include a body that contains a status information of the created job that conforms to the statusInfo.yaml schema.
B	The response SHALL contain a `created` status code and the `id` property that contains the job identifier.

6.2.5. Exceptions

See Clause 6.1.1 for general guidance.

If the request body's media type is not supported by the server, see requirement /req/deploy-replace-undeploy/deploy/unsupported-media-type from OGC 20-044.

REQUIREMENT 7	
LABEL	/req/job-management/create-unsupported-schema
A	If the server does not support the Content-Schema header associated with the request body, the code of the response SHALL be `422 Unprocessable Content`.
B	The content of that response SHALL be based upon the OpenAPI 3.0 schema exception.yaml.
C	The `type` of the exception SHALL be "http://www.opengis.net/def/exceptions/ogcapi-processes-4/1.0/unsupported-schema".

6.3. Updating an existing job

6.3.1. Sequence diagram

The following diagram illustrates the sequence diagram for updating a previously created job. The identifier of the job does not change.

NOTE: The new job definition replaces the old job definition. Version control is not discussed in this Standard.

```
Client                                                              Server
  |                                                                   |
  |   PATCH /jobs/{jobID}   HTTP/1.1                                  |
  |   Content-Type: application/json                                  |
  |                                                                   |
  |------------------------------------------------------------->    |
  |                                                                   |
  |   HTTP/1.1 200 OK                                                 |
  |<-------------------------------------------------------------    |
```

Listing 2

6.3.2. Operation

REQUIREMENT 8	
LABEL	/req/job-management/update-patch-op
A	For every created jobs (path `/jobs/{jobID}`), the server SHALL support the HTTP PATCH operation.
B	The parameter `jobID` is each `id` property in the jobs list response (JSONPath: `$.jobs[*].id`).

6.3.3. Request body

6.3.4. Overview

REQUIREMENT 9	
LABEL	/req/job-management/update-body

REQUIREMENT 9	
A	The body of a PATCH request SHALL be in JSON format.

PERMISSION 4	
LABEL	/per/job-management/update-body
A	A server MAY support any encoding in the body of a HTTP PATCH operation.

REQUIREMENT 10	
LABEL	/req/job-management/update-content-type
A	As per HTTP 1.1 (https://tools.ietf.org/html/rfc2616#section-14.17) the 'Content-Type' header SHALL be used to indicate the media type of a request body.

PERMISSION 5	
LABEL	/per/job-management/update-content-schema
A	The `Content-Schema` header MAY be used to indicate the schema of a request body for updating a job.

6.3.4.1. OGC API — Processes — Workflow Execute Request

RECOMMENDATION 3	
LABEL	/rec/job-management/update-body-ogcapi-processes
A	If a job can be created from an OGC API — Processes — Workflow Execute Request, implementations SHOULD consider supporting the OGC API — Processes — Workflow Execute Request encoding.

6.3.4.2. OpenEO Process Graph

RECOMMENDATION 4	
LABEL	/rec/job-management/update-body-openeo
A	If a job can be created from an OpenEO Process Graph, implementations SHOULD consider supporting the OpenEO Process Graph encoding.

6.3.5. Response

REQUIREMENT 11	
LABEL	/req/job-management/update-response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200` or `204`.

The status code depends on the server. If the server has replaced the job, the response is either `200` (if the response includes additional content) or `204` (if the response has no additional content).

6.3.6. Exceptions

See Clause 6.1.1 for general guidance.

If the request body's media type is not supported by the server, see requirement /req/deploy-replace-undeploy/deploy/unsupported-media-type from OGC 20-044.

If the job with the specified identifier does not exist on the server, see requirement /req/core/job-exception-no-such-job from OGC 18-062r2.

REQUIREMENT 12	
LABEL	/req/job-management/update-response-locked
A	If a job is locked, meaning that it is currently being processed (status set to `accepted` or `running`), the server SHALL respond with HTTP status code `423 Locked`.
B	The response body SHALL be based upon the OpenAPI 3.0 schema exception.yaml.
C	The `type` of the exception SHALL be "http://www.opengis.net/def/exceptions/ogcapi-processes-4/1.0/locked".

6.4. Staring a job

6.4.1. Sequence diagram

The following diagram illustrates the sequence diagram for starting the execution of a previously created jobs.

```
Client                                                        Server
  |                                                              |
  |  POST /jobs/{jobID}/results   HTTP/1.1                       |
  |------------------------------------------------------------->|
  |                                                              |
  |  HTTP/1.1 200 OK                                             |
  |<-------------------------------------------------------------|
```

Listing 3

6.4.2. Operation

REQUIREMENT 13	
LABEL	/req/job-management/start-post-op
A	For every created jobs (path: `/jobs/{jobID}/results`), the server SHALL support the HTTP POST operation.
B	The parameter `jobID` is each `id` property in the job list response (JSONPath: `$.jobs[*].id`).

6.4.3. Response

REQUIREMENT 14	
LABEL	/req/job-management/start-response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code ‘200’.
B	A response SHALL be a document that conforms to statusInfo.yaml.

6.4.4. Exceptions

See HTTP status codes for general guidance.

If the job with the specified identifier does not exist on the server, see requirement /req/core/job-exception-no-such-job from OGC 18-062r2.

If the job with the specified identifier is locked, see requirement /req/job-management/update/response-locked from Clause 6.3.

6.5. Job definition

For every job, it is possible to retrieve its original definition. It corresponds to the request's body used to create or update the jobs.

6.5.1. Operation

REQUIREMENT 15	
LABEL	/req/job-management/definition-get-op
A	For every jobs (using method: POST on path: /jobs/{jobID}), the server SHALL support the HTTP GET operation at the path `/jobs/{jobID}/definition`.
B	The parameter `jobID` is each `id` property in the job-list response (JSONPath: `$.jobs[*].id`).

6.5.2. Response

6.5.2.1. Overview

REQUIREMENT 16	
LABEL	/req/job-management/definition-response-success
A	A successful access to the resource SHALL be reported as a response with a HTTP status code `200`.

REQUIREMENT 17	
LABEL	/req/job-management/definition-response-body
A	A response with HTTP status code `200` SHALL include a body that contains the request body used to create or update the job.

6.5.3. Exceptions

See HTTP status codes for general guidance.

If the job with the specified identifier does not exist on the server, see requirement /req/core/job-exception-no-such-job from OGC 18-062r2.

7

REQUIREMENTS CLASS “OGC API — PROCESS — WORKFLOW EXECUTE REQUEST”

7 REQUIREMENTS CLASS “OGC API — PROCESS — WORKFLOW EXECUTE REQUEST”

7.1. Overview

REQUIREMENTS CLASS 2	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITES	OGC API — Processes — Part 1: Core OGC API — Processes — Part 3: Workflows, Conformance Class ‘nested-processing’ http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/job-management
LABEL	http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/ogcapi-processes

This requirements class defines that the OGC API — Process — Workflow Execute Request is supported as an encoding for job definitions.

7.2. OGC API — Processes — Workflow Execute Request

7.2.1. Overview

REQUIREMENT 18	
LABEL	/req/ogcapi-processes/schema
A	An `OGC API - Processes - Workflow - Execute Request` document SHALL be based upon the JSON schema execute-workflow.yaml.

7.3. Creating a new job

7.3.1. Request body

REQUIREMENT 19	
LABEL	/req/ogcapi-processes/create-body
A	The body of the POST request SHALL be based upon the OpenAPI 3.0 schema execute-workflows.yaml
B	The media type `application/json` SHALL be used to indicate that request body contains a processes description encoded as an OGC API — Processes.

PERMISSION 6	
LABEL	/per/ogcapi-processes/create-content-schema
A	The `Content-Schema` header MAY be pointing to the OpenAPI 3.0 schema execute-workflows.yaml.

7.4. Updating an existing job

7.4.1. Request body

REQUIREMENT 20	
LABEL	/req/ogcapi-processes/update-body
A	The media type `application/ogcapi-processes+json` SHALL be used to indicate that request body contains a job encoded as an OpenEO.

PERMISSION 7	
LABEL	/per/ogcapi-processes/update-content-schema

PERMISSION 7	
A	The `Content-Schema` header MAY be pointing to the OpenAPI 3.0 schema execute-workflows.yaml.

7.5. Job definition

7.5.1. Response content

REQUIREMENT 21	
LABEL	/req/ogcapi-processes/definition-response-body
A	A response with HTTP status code `200` SHALL include a body that contains the OGC API — Processes — Workflow — Execute Request to use to deploy the process.

8

REQUIREMENTS CLASS “OPENEO PROCESS GRAPH”

8 REQUIREMENTS CLASS “OPENEO PROCESS GRAPH”

8.1. Overview

REQUIREMENTS CLASS 3	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/job-management
LABEL	http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/openeo

This requirements class defines that the server supports the OpenEO Process Graph as an encoding for job definitions.

8.2. OpenEO Process Graph

8.2.1. Overview

REQUIREMENT 22	
LABEL	/req/openeo/schema
A	An `OpenEO Process Graph` document SHALL be based upon the OpenEO Process Graph JSON schema https://openeo.org/documentation/1.0/developers/api/assets/pg-schema.json.

```
type: object
additionalProperties:
  type: object
  required:
    - process_id
    - arguments
  properties:
    process_id:
      type: string
    arguments: {}
```

Listing 4 — Schema for OpenEO Process Graph

8.3. Creating a new job

8.3.1. Request body

REQUIREMENT 23	
LABEL	/req/openeo/create-body
A	The media type `application/json` SHALL be used to indicate that request body contains a processes description encoded as an OpenEO Process Graph.

PERMISSION 8	
LABEL	/per/openeo/create-content-schema
A	The `Content-Schema` header MAY be pointing to OpenEO Process Graph schema.

8.4. Updating an existing job

8.4.1. Request body

REQUIREMENT 24	
LABEL	/req/openeo/update-body
A	The media type `application/json` SHALL be used to indicate that request body contains a job encoded as an OpenEO Process Graph.

PERMISSION 9	
LABEL	/per/openeo/update-content-schema
A	The `Content-Schema` header MAY be pointing to OpenEO Process Graph schema.

8.5. Job definition

8.5.1. Response content

REQUIREMENT 25	
LABEL	/req/openeo/definition-response-body
A	A response with HTTP status code `200` SHALL include a body that contains the OpenEO Process Graph to use to deploy the process.

9

REQUIREMENTS CLASS “PROVENANCE”

9 REQUIREMENTS CLASS “PROVENANCE”

9.1. Overview

This requirements class defines how to allow client application accessing the provenance of a job run.

REQUIREMENTS CLASS 4	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	OGC API — Processes — Part 1: Core
LABEL	http://www.opengis.net/spec/ogcapi-processes-4/1.0/req/provenance

9.2. Additional endpoints

9.2.1. Inputs

The server MUST provide an endpoint to retrieve the inputs of a job run.

9.2.1.1. Operation

REQUIREMENT 26	
LABEL	/req/provenance/inputs-get-op
A	For every created jobs (path: `/jobs/{jobID}/inputs`), the server SHALL support the HTTP GET operation.
B	The parameter `jobID` is each `id` property in the job list response (JSONPath: `$.jobs[*].id`).

9.2.1.2. Response

REQUIREMENT 27	
LABEL	/req/provenance/inputs-response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code '200'.
B	The response SHALL contains a JSON document that conforms to the schema inputs.yaml.

9.2.2. Prov

The server MUST provide an endpoint to retrieve the provenance of a job.

9.2.2.1. Operation

REQUIREMENT 28	
LABEL	/req/provenance/prov-get-op
A	For every created jobs (path: `/jobs/{jobID}/prov`), the server SHALL support the HTTP GET operation.
B	The parameter `{jobID}` is each `id` property in the job list response (JSONPath: `$.jobs[*].id`).

PERMISSION 10	
LABEL	/per/provenance/prov-content-negotiation
A	Content negotiation MAY be supported to provide alternate representations of the response.
B	The server MAY support the following additional content type: `application/ld+json` for PROV-O as JSON-LD
C	The server MAY support the following additional content type: `application/xml` for PROV-XML
D	The server MAY support the following additional content type: `text/provenance-notation; charset="UTF-8"` for PROV-N.

9.2.2.2. Response

REQUIREMENT 29	
LABEL	/req/provenance/prov-response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code ‘200’.
B	Per default, the response SHALL contains a JSON document that conforms to the schema for PROV-JSON.
C	In case content-negotiation is used, the response MAY contain other representations including PROV-O as JSON-LD, PROV-XML or PROV-N.

10 OPENAPI 3.0

10 OPENAPI 3.0

See OGC 18-062r2, Clause 9.

11

MEDIA TYPES

11 MEDIA TYPES

See OGC 18-062r2, Clause 13.

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A.1. Introduction

OGC Web Application Programming Interfaces (APIs) are not Web Services in the traditional sense. Rather, they define the behavior and content of a set of Resources exposed through a Web API. Therefore, an API endpoint may expose resources in addition to those defined by the standard. A test engine must be able to traverse an implementation of the API, identify and validate test points, and ignore resource paths which are not to be tested.

The Web API under test can require authorization. Any Executable Test Suite implementing this test suite should implement the following security schemes supported by OpenAPI 3.0: HTTP Authorization schemes “basic” and “bearer”, API keys, and OAuth2 flow “authorizationCode”.

A.2. Conformance Class Job Management

CONFORMANCE CLASS A.1: JOB MANAGEMENT	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-processes-4/1.0/conf/job-management`
SUBJECT	http://www.opengis.net/spec/ogcapi-processes-4/1.0/conf/job-management
TARGET TYPE	Web API
CONFORMANCE TEST	Conformance test A.1-1: `/conf/dru/deploy/post-op`

A.2.1. Create operation

ABSTRACT TEST A.1	
IDENTIFIER	`/conf/jm/create/post-op`
REQUIREMENT	/req/job-management/create-post-op
TEST PURPOSE	Validate that the server support HTTP POST operation at the path /jobs/
TEST METHOD	1. Construct a path for the {root}/jobs path. 2. Issue an HTTP POST request for each path. 3. Validate that the response header does not contain `405 Method not allowed`

B

ANNEX B (INFORMATIVE) REVISION HISTORY

B

ANNEX B (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2024-08-22	None	Gérald Fenoy	all	Boostraping the document

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] OpenAPI Initiative. OpenAPI Specification 3.0.2. Available at: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md.

[2] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić (editors), Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, Luka Stojanovic (2020): Common Workflow Language, v1.2. Specification, Common Workflow Language working group. https://w3id.org/cwl/v1.2/

[3] OpenEO: OpenEO Developers API Reference / Process Graphs. https://openeo.org/documentation/1.0/developers/api/reference.html#section/Processes/Process-Graphs