OGC TESTBED 19 DRAFT API - GEODATACUBES SPECIFICATION

ENGINEERING REPORT

DRAFT

Submission Date: 2024-03-05
Approval Date: TBD
Publication Date: TBD
Editor: Matthias Mohr

Notice: This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.
License Agreement
Use of this document is subject to the license agreement at https://www.ogc.org/license

Copyright notice
Copyright © 2024 Open Geospatial Consortium
To obtain additional rights of use, visit https://www.ogc.org/legal

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

I. EXECUTIVE SUMMARY ........................................................................................................................................iv
II. KEYWORDS .........................................................................................................................................................iv
III. CONTRIBUTORS ..................................................................................................................................................iv
1. INTRODUCTION ..................................................................................................................................................7
2. GEODATACUBE API DRAFT SPECIFICATION ...............................................................................................9
ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS ..................................................................................12
ANNEX B (NORMATIVE) GEODATACUBE API AS OPENAPI SPECIFICATION .............................14

LIST OF FIGURES

Figure B.1 ..............................................................................................................................................................14
EXECUTIVE SUMMARY

This OGC Testbed 19 Engineering Report documents a draft OGC API — GeoDataCube Standard (aka GDC API). The OGC Member participants in this Testbed 19 activity developed, documented, and tested the draft OGC GDC API Standard. The draft will be submitted to the OGC GeoDataCube Standards Working Group (SWG) as a new standards work item.

The OGC GeoDataCube SWG was chartered to respond to the long-standing issue of establishing a standard that supports accessing and processing geospatial datacubes in an interoperable way. The draft OGC API — GeoDataCube was developed in OGC Testbed 19 responds to this need and proposes a draft API specification.

The Testbed 19 GDC initiative targeted enhanced interoperability. The draft GDC API Standard based on of OGC API — Common, OGC API — Coverages Standard, OGC API — Processes Standard, the STAC API, and the openEO API. The Testbed 19 participants concentrated on server and client application development, and usability testing based on conformance classes and use cases. The draft GDC API is defined as an OpenAPI 3.0 document and provides endpoints for capabilities, data discovery/access, process discovery, and data processing. Notably, the draft GDI API Standard is extensible through additional implementations of OGC API Standards or openEO API parts. Documentation is available in machine-readable YAML and human-friendly HTML through a GitHub repository.

NOTE: In this document, any occurrence of the phrase “GDC API” means and can be expanded to “draft OGC API — GeoDataCube Standard”.

KEYWORDS

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

gеographic, data cubes, api

CONTRIBUTORS

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>ORGANIZATION</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthias Mohr</td>
<td>Eurac Research</td>
<td>Editor</td>
</tr>
</tbody>
</table>

OPEN GEOSPATIAL CONSORTIUM 23-048
<table>
<thead>
<tr>
<th>NAME</th>
<th>ORGANIZATION</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander Jacob</td>
<td>Eurac Research</td>
<td>Contributor</td>
</tr>
</tbody>
</table>
INTRODUCTION
INTRODUCTION

Over the past decade, GeoDataCubes were developed independently, resulting in a lack of interoperability between different implementations. By improving interoperability, the vendor community will be able to proceed with specific GeoDataCube variants that meet specific community requirements. At the same time the consumer community will be able to interact much more effectively with different implementation instances.

The OGC Geodatacube Standards Working Group was formed in 2023 and the Testbed 19 work was designed to provide initial input to the work of the SWG. Testbed 19 focused on the development of a draft GeoDataCube API, and the development of a number of client and server applications for both data access, visualization and processing. Three use cases were used to test the implementations. Usability tests ensured that the draft GeoDataCube API Standard deployed in developed software was user-friendly. See the corresponding Engineering Report for details about client implementations, server implementations, use cases, and usability tests.
2

GEODATACUBE API DRAFT SPECIFICATION
The draft GDC API Standard is defined as a RESTful Web API utilizing JSON and HTTP that provides access to geospatial data cubes and their related metadata. The draft is based on multiple other API standards and specifications that provide the building blocks for the GDC API. The following list provides a brief description of the building blocks:

- OGC API — Common — Part 1 & 2 (required)
- OGC API — Coverages — Part 1 (required)
- OGC API — Processes — Part 1 (optional)
- STAC API, which is based on OGC API — Features — Part 1 (optional)
- openEO API (optional)

The following list groups the GDC API endpoints and maps them to the OGC, STAC, and openEO building blocks they are based on:

- Capabilities: OGC API — Common — Part 1, openEO API
- Process Discovery: OGC API — Processes — Part 1, openEO API
- Data Processing: OGC API — Processes — Part 1, openEO API

Please note that although most of the documents listed above are published and stable specifications or standards, the OGC API — Coverages standard is in draft and will likely change before its final publication.

The GDC API document highlights whenever two API building blocks, e.g., from the openEO API and OGC API — Processes, share the same endpoint and explains how the endpoints can be combined. The draft standard also provides information how to distinguish the elements returned by an API endpoint so that they can be identified as belonging to one of the respective building blocks.

The GDC API can be extended with additional functionality by implementing additional parts of the OGC API Standards suite or the openEO API.

The GDC API is specified using the OpenAPI 3.0 standard. The specification document can be found here in machine-readable format (OpenAPI 3.0, YAML):

- https://raw.githubusercontent.com/m-mohr/geodatacube-api/master/openapi.yaml
- or in Appendix B of this document
The draft GDC API is also available rendered as HTML in a more human-friendly format:

- https://m-mohr.github.io/geodatacube-api/

The GitHub repository that contains the GDI API is available here:

- https://github.com/m-mohr/geodatacube-api/
ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS
## ANNEX A
(NORMATIVE)
ABBREVIATIONS/ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>STAC</td>
<td>SpatioTemporal Asset Catalog</td>
</tr>
<tr>
<td>YAML</td>
<td>Yet Another Markup Language</td>
</tr>
</tbody>
</table>
ANNEX B (NORMATIVE) GEODATACUBE API AS OPENAPI SPECIFICATION
ANNEX B
(NORMATIVE)
GEODATACUBE API AS OPENAPI SPECIFICATION

openapi: 3.0.2
info:
  title: geodatacube API
  version: 1.0.0-beta
  description: |
      The geodatacube API specification for interoperable cloud-based processing
      of large Earth observation datacubes.

  **Conformance class**: `https://api.geodatacube.example/1.0.0-beta`

# API Principles

## Language

In the specification the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119](https://www.rfc-editor.org/rfc/rfc2119.html) and [RFC 8174](https://www.rfc-editor.org/rfc/rfc8174.html).

### Casing

Unless otherwise stated the API works **case sensitive**.

All names SHOULD be written in snake case, i.e. words are separated with one underscore character (\_) and no spaces, with all letters lowercased. Example: `hello_world`. This applies particularly to endpoints and JSON property names. HTTP header fields are generally case-insensitive according to [RFC 7230](https://www.rfc-editor.org/rfc/rfc7230.html#section-3.2) and in the specification we follow their respective casing conventions, e.g. `Content-Type` or `GDC-Identifier`, for better readability and consistency.

### HTTP / REST

This uses [HTTP REST](https://en.wikipedia.org/wiki/Representational_state_transfer) [Level 2](https://martinfowler.com/articles/richardsonMaturityModel.html#level2) for communication between client and back-end server.

Public APIs MUST be available via HTTPS only.

Endpoints are made use meaningful HTTP verbs (e.g. GET, POST, PUT, PATCH, DELETE) whenever technically possible. If there is a need to transfer big chunks of data for a GET requests to the back-end, POST requests MAY be used as a replacement as they support to send data via request body. Unless otherwise stated, PATCH requests are only defined to work on direct (first-level) children of the full JSON object. Therefore, changing a property on a
deeper level of the full JSON object always requires to send the whole JSON object defined by the first-level property.

Naming of endpoints follow the REST principles. Therefore, endpoints are centered around resources. Resource identifiers MUST be named with a noun in plural form except for single actions that can not be modelled with the regular HTTP verbs. Single actions MUST be single endpoints with a single HTTP verb (POST is RECOMMENDED) and no other endpoints beneath it.

The API makes use of [HTTP Content Negotiation](https://www.rfc-editor.org/rfc/rfc9110.html#name-content-negotiation), including, but not limited to, the request headers `Accept`, `Accept-Charset` and `Accept-Language`.

### JSON

The API uses JSON for request and response bodies whenever feasible. Services use JSON as the default encoding. Other encodings can be requested using HTTP Content Negotiation ([`Accept` header](https://www.rfc-editor.org/rfc/rfc9110.html#name-accept)). Clients and servers MUST NOT rely on the order in which properties appear in JSON. To keep the response size small, lists of resources (e.g. the list of batch jobs) usually should not include nested JSON objects, if this information can be requested from the individual resource endpoints (e.g. the metadata for a single batch job).

### Charset

Services use [UTF-8](https://en.wikipedia.org/wiki/UTF-8) as the default charset if not negotiated otherwise with HTTP Content Negotiation ([`Accept-Charset` header](https://www.rfc-editor.org/rfc/rfc9110.html#name-accept-charset)).

### Web Linking

The API is designed in a way that to most entities (e.g. collections and processes) a set of links can be added. These can be alternate representations, e.g. data discovery via OGC WCS or OGC CSW, references to a license, references to actual raw data for downloading, detailed information about pre-processing and more. Clients should allow users to follow the links.

Whenever links are utilized in the API, the description explains which relation (`rel` property) types are commonly used.

A [list of standardized link relations types is provided by IANA](https://www.iana.org/assignments/link-relations/link-relations.xhtml) and the API tries to align whenever feasible.

Some very common relation types - usually not mentioned explicitly in the description of `links` fields - are:

1. `self`: which allows link to the location that the resource can be (permanently) found online. This is particularly useful when the data is data is made available offline, so that the downstream user knows where the data has come from.

2. `alternate`: An alternative representation of the resource, may it be another metadata standard the data is available in or simply a human-readable version in HTML or PDF.

3. `about`: A resource that is related or further explains the resource, e.g. a user guide.
4. `canonical`: This relation type usually points to a publicly accessible and more long-lived URL for a resource that otherwise often requires (Bearer) authentication with a short-lived token. This way the the exposed resources can be used by clients without additional authentication steps.

   For example, a shared user-defined process or batch job results could be exposed via a canonical link. If a URL should be publicly available to everyone, it can simply a user-specific URL, e.g. `https://geodatacube.example/processes/john_doe/ndvi`.

   For resources that should only be accessible to a certain group of user, a signed URL could be given, e.g. `https://geodatacube.example/processes/81zjh1tc2pt52gbx/ndvi`.

   Generally, it is RECOMMENDED to add descriptive titles (property `title`) and media type information (property `type`) for a better user experience.

## Error Handling

The success of requests MUST be indicated using [HTTP status codes](https://www.rfc-editor.org/rfc/rfc7231.html#section-6) according to [RFC 7231](https://www.rfc-editor.org/rfc/rfc7231.html).

If the API responds with a status code between 100 and 399 the back-end indicates that the request has been handled successfully.

In general an error is communicated with a status code between 400 and 599. Client errors are defined as a client passing invalid data to the service and the service *correctly* rejecting that data. Examples include invalid credentials, incorrect parameters, unknown versions, or similar. These are generally "4xx" HTTP error codes and are the result of a client passing incorrect or invalid data. Client errors do *not* contribute to overall API availability.

Server errors are defined as the server failing to correctly return in response to a valid client request. These are generally "5xx" HTTP error codes. Server errors *do* contribute to the overall API availability. Calls that fail due to rate limiting or quota failures MUST NOT count as server errors.

### JSON error object

A JSON error object SHOULD be sent with all responses that have a status code between 400 and 599.

```json
{
  "id": "936DA01F-9ABD-4D9D-80C7-02AF85C822A8",
  "code": "SampleError",
  "message": "A sample error message.",
  "url": "https://geodatacube.example/docs/errors/SampleError"
}
```

Sending `code` and `message` is REQUIRED.

* A back-end MAY add a free-form `id` (unique identifier) to the error response to be able to log and track errors with further non-disclosable details.
* The `code` is proprietary textual error code.
* The `message` explains the reason the server is rejecting the request. For "4xx" error codes the message explains how the client needs to modify the request.
By default the message MUST be sent in English language. Content Negotiation is used to localize the error messages: If an `Accept-Language` header is sent by the client and a translation is available, the message should be translated accordingly and the `Content-Language` header must be present in the response. See "[How to localize your API](http://apiux.com/2013/04/25/how-to-localize-your-api/)" for more information.

* `url` is an OPTIONAL attribute and contains a link to a resource that is explaining the error and potential solutions in-depth.

### Standardized status codes

The API usually uses the following HTTP status codes for successful requests:

- **200 OK**: Indicates a successful request **with** a response body being sent.
- **201 Created**: Indicates a successful request that successfully created a new resource. Sends a `Location` header to the newly created resource **without** a response body.
- **202 Accepted**: Indicates a successful request that successfully queued the creation of a new resource, but it has not been created yet. The response is sent **without** a response body.
- **204 No Content**: Indicates a successful request **without** a response body being sent.

The API has some commonly used HTTP status codes for failed requests:

- **400 Bad Request**: The back-end responds with this error code whenever the error has its origin on client side and no other HTTP status code in the 400 range is suitable.
- **401 Unauthorized**: The client did not provide any authentication details for a resource requiring authentication or the provided authentication details are not correct.
- **403 Forbidden**: The client did provided correct authentication details, but the privileges/permissions of the provided credentials do not allow to request the resource.
- **404 Not Found**: The resource specified by the path does not exist, i.e. one of the resources belonging to the specified identifiers are not available at the back-end.
  
  *Note:* Unsupported endpoints MAY also return HTTP status code 501.

- **500 Internal Server Error**: The error has its origin on server side and no other status code in the 500 range is suitable.

- **501 Not Implemented**: The requested endpoint is part of the API specification, but is not implemented (yet) by the back-end.
  
  *Note:* Unsupported endpoints MAY also return HTTP status code 404.

If a HTTP status code in the 400 range is returned, the client SHOULD NOT repeat the request without modifications. For HTTP status code in the 500 range, the client MAY repeat the same request later.
All HTTP status codes defined in RFC 7231 in the 400 and 500 ranges can be used as error code in addition to the most used status codes mentioned here. Responding with error codes 400 and 500 SHOULD be avoided in favor of any more specific standardized or proprietary error code.

## Temporal data

Date, time, intervals and durations are formatted based on ISO 8601 or its profile [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) whenever there is an appropriate encoding available in the standard. All temporal data are specified based on the Gregorian calendar.

# Authentication

The API offers one form of authentication by default:
* Basic at `GET /credentials/basic`
* OpenID Connect at `GET /credentials/oidc`

After authentication with any of the methods listed above, the tokens obtained during the authentication workflows can be sent to protected endpoints in subsequent requests.

Further authentication methods MAY be added by back-ends.

<SecurityDefinitions />

**Note:** Although it is possible to request several public endpoints for capabilities and discovery that don't require authorization, it is RECOMMENDED that clients (re-)request the public endpoints that support Bearer authentication with the Bearer token once available to also retrieve any private data that is made available specifically for the authenticated user. This may require that clients clear any cached data they retrieved from public endpoints before.

## Cross-Origin Resource Sharing (CORS)

> Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources [...] on a web page to be requested from another domain outside the domain from which the first resource was served. [...] 
> CORS defines a way in which a browser and server can interact to determine whether or not it is safe to allow the cross-origin request. It allows for more freedom and functionality than purely same-origin requests, but is more secure than simply allowing all cross-origin requests.

Source: [https://en.wikipedia.org/wiki/Cross-origin_resource_sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing)

Geodatacube-API-based back-ends are usually hosted on a different domain / host than the client that is requesting data from the back-end. Therefore most requests to the back-end are blocked by all modern browsers. This leads to the problem that the JavaScript library and any browser-based application can't access back-ends. Therefore, all back-end providers SHOULD support CORS to enable browser-based applications to access back-ends. [CORS is a recommendation of the W3C organization](https://www.w3.org/TR/cors/). The following chapters will explain how back-end providers can implement CORS support.

**Tip:** Most servers can send the required headers and the responses to the OPTIONS requests automatically for all endpoints. Otherwise you may also use a proxy server to add the headers and OPTIONS responses.

## CORS headers
The following headers MUST be included with every response:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Access-Control-Allow-Origin</td>
<td>Allowed origin for the request, including protocol, host and port or <code>*</code> for all origins. It is RECOMMENDED to return the value <code>*</code> to allow requests from browser-based implementations.</td>
</tr>
<tr>
<td>Access-Control-Expose-Headers</td>
<td>Some endpoints require to send additional HTTP response headers such as 'GDC-Identifier' and 'Location'. To make these headers available to browser-based clients, they MUST be white-listed with this CORS header. The following HTTP headers are white-listed by browsers and MUST NOT be included: 'Cache-Control', 'Content-Language', 'Content-Length', 'Content-Type', 'Expires', 'Last-Modified' and 'Pragma'. At least the following headers MUST be listed in this version of the API: 'Link', 'Location', and 'GDC-Identifier'.</td>
</tr>
</tbody>
</table>

### Example request and response

Request:
```
```

Response:
```
```

## OPTIONS method

All endpoints must respond to the `OPTIONS` HTTP method. This is a response for the preflight requests made by web browsers before sending the actual request (e.g. `POST /jobs`). It needs to respond with a status code of `204` and no response body.

**In addition** to the HTTP headers shown in the table above, the following HTTP headers MUST be included with every response to an OPTIONS request:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Access-Control-Allow-Headers</td>
<td>Comma-separated list of HTTP headers allowed to be sent with the actual (non-preflight) request. MUST contain at least 'Authorization' if any kind of authorization is implemented by the back-end.</td>
</tr>
</tbody>
</table>
### Example request and response

Request:
```
```
OPTIONS /api/v1/jobs HTTP/1.1
Host: geodatacube.example
Origin: https://company.example:8080
Access-Control-Request-Method: POST
Access-Control-Request-Headers: Authorization, Content-Type
```
```
Note that the `Access-Control-Request-*` headers are automatically
attached to the requests by the browsers.
```
```
Response:
```
```
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: OPTIONS, GET, POST, PATCH, PUT, DELETE
Access-Control-Allow-Headers: Authorization, Content-Type
Access-Control-Expose-Headers: Location, GDC-Identifier, Link
Content-Type: application/json
```
```
O
P
E
N
G
E
O
S
T
A
C
I
O
N
S
O
R
I
M
2
3
0
4
8
```
* accounting related tasks, e.g. processing costs and creating invoices,
* user registration (except for [user registration with OpenID Connect](http://openid.net/specs/openid-connect-registration-1_0.html)).

- name: Data Discovery / Access
description: |

These endpoints allow to list the collections that are available at the back-end and can be used as data cubes for data processing. It builds on top of:

- OGC API - Coverages - Part 1
- STAC API (incl. STAC Data Cube extension)
- name: OGC API - Coverages
description: Data access through OGC API - Coverages - Part 1 (v0.0.2)
- name: OGC API - Features / STAC API
description: >-
  **OPTIONAL.** Data access through OGC API - Features - Part 1 (v1.0.0)
and STAC API (v1.0.0)
- name: Process Discovery
description: |

**OPTIONAL.** These endpoints allow to list the predefined processes that are available at the back-end. To list user-defined processes see 'openEO - User-Defined Processes'.
- name: OGC API - Processes
description: >-
  **OPTIONAL.** Data processing through OGC API - Processes - Part 1 (v1.0.0)
- name: openEO
description: >-
  **OPTIONAL.** Data processing through openEO (v1.2.0)
- name: openEO - User-Defined Processes
description: >-
  **OPTIONAL.** These endpoints allow to store and manage user-defined processes with their process graphs at the back-end.
- name: openEO - Secondary Services (OGC APIs)
description: >-
  **OPTIONAL.** On-demand access to data using other web service protocols (e.g., OGC API - Tiles / Maps).

servers:
- url: 'https://geodatacube.example/api'
description: >-
The URL of the API MAY freely be chosen by the back-end providers. Nevertheless, all servers MUST support HTTPS as the authentication methods
are not secure with HTTP only!

paths:
/:
  get:
    summary: Information about the back-end
    operationId: capabilities
description: >-
    Lists general information about the back-end, including which version and endpoints of the geodatacube API are supported. May also include.

billing
  information.
tags:
  - Capabilities
security:
  - {}
responses:
'200':
description: |
  Information about the API version and supported endpoints / features.
This endpoint MUST always be available for the API to be valid.

content:
application/json:
  schema:
    title: Capabilities
    type: object
    required:
    - id
    - title
    - description
    - gdc_version
    - endpoints
    - links
    properties:
      gdc_version:
        type: string
        description: Version number of the geodatacube API specification this back-end implements.
        enum:
        - 1.0.0-beta
      backend_version:
        type: string
        description: Version number of the back-end implementation.
      example: 1.1.2
      stac_version:
        $ref: '#/components/schemas/stac_version'
      api_version:
        type: string
        description: If the openEO API is implemented: Version number of the openEO API specification this back-end implements.
        example: 1.2.0
      type:
        type: string
        enum:
        - Catalog
        description: For STAC versions >= 1.0.0-rc.1 this field is required.
        example: Catalog
      id:
        type: string
        description: Identifier for the service.
        example: cool-eo-cloud
      title:
        type: string
        description: The name of the service.
        example: Example Cloud Corp.
      description:
        type: string
        format: commonmark
        description: This field originates from STAC and is used as unique identifier for the STAC catalog available at `collections`.
        example: cool-eo-cloud
A description of the service, which allows the service provider to introduce the user to its service.

[CommonMark 0.29](http://commonmark.org/) syntax MAY be used for rich text representation.

eexample:  
This service is provided to you by [Example Cloud Corp.](https://cloud.example). It implements the full geodatacube API and allows to process a range of 999 EO data sets, including

* Sentinel 1/2/3 and 5
* Landsat 7/8

A free plan is available to test the service. For further information please contact our customer service at [support@cloud.example](mailto:support@cloud.example).

conformsTo:  
$ref: `#/components/schemas/conformsTo`
endpoints:  
type: array

description: >-
Lists all supported endpoints. Supported are all endpoints, which are implemented, return usually a 2XX or 3XX HTTP status code and are fully compatible to the API specification. An entry for this endpoint (path `/` with method ``GET`) SHOULD NOT be listed. Each path MUST only be listed once in the array.

items:  
title: Endpoint

type: object

required:  
- path
- methods

properties:  
path:  
description: >-
Path to the endpoint, relative to the URL of this endpoint. In general the paths MUST follow the paths specified in the openAPI specification as closely as possible. Therefore, paths MUST be prepended with a leading slash, but MUST NOT contain a trailing slash. Variables in the paths MUST be placed in curly braces and follow the parameter names in the openAPI specification, e.g. `/{job_id}`.


type: string

methods:  
description: >-
Supported HTTP verbs in uppercase. It is OPTIONAL to list `OPTIONS` as method (see the [CORS section](#section/Cross-Origin-Resource-Sharing-(CORS))).


type: array

items:  
type: string

enum:
- GET
- POST
- PATCH
- PUT
- DELETE
- OPTIONS

example:
- path: /collections
  methods:
    - GET
- path: '/collections/{collection_id}'
  methods:
    - GET
- path: /processes
  methods:
    - GET
- path: /jobs
  methods:
    - GET
    - POST
- path: '/jobs/{job_id}'
  methods:
    - GET
    - DELETE
    - PATCH

links:

description: |-
  Links related to this service, e.g. the homepage of the service provider or the terms of service.

  1. `terms-of-service` (optional): A link to the terms of service. If a back-end provides a link to the terms of service, the clients MUST provide a way to read the terms of service and only connect to the back-end after the user agreed to them. The user interface MUST be designed in a way that the terms of service are not agreed to by default, i.e. the user MUST explicitly agree to them.

  2. `privacy-policy` (optional): A link to the privacy policy (GDPR). If a back-end provides a link to a privacy policy, the clients MUST provide a way to read the privacy policy and only connect to the back-end after the user agreed to them. The user interface MUST be designed in a way that the privacy policy is not agreed to by default, i.e. the user MUST explicitly agree to them.

  3. `service-desc` (required) and `service-doc` (optional): A link to the API definition. Use `service-desc` for machine-readable API definition and `service-doc` for human-readable API definition.

  4. `http://www.opengis.net/def/rel/ogc/1.0/conformance` (required): A link to the Conformance declaration (see `/conformance`).

  5. `data` (required): A link to the collections (see `'/collections'`).

  6. `create-form` (optional): A link to a user registration page.

  7. `recovery-form` (optional): A link to a page where a user can
recover a user account (e.g. to reset the password or send a reminder about the username to the user's email account).

For additional relation types see also the lists of [common relation types](#section/API-Principles/Web-Linking).

type: array
items:
$ref: '#/components/schemas/link'
example:
- href: 'https://cloud.example'
  rel: about
  type: text/html
  title: Homepage of the service provider
- href: 'https://cloud.example/tos'
  rel: terms-of-service
  type: text/html
  title: Terms of Service
- href: 'https://cloud.example/privacy'
  rel: privacy-policy
  type: text/html
  title: Privacy Policy
- href: 'https://cloud.example/register'
  rel: create-form
  type: text/html
  title: User Registration
- href: 'https://cloud.example/lost-password'
  rel: recovery-form
  type: text/html
  title: Reset Password
- href: 'https://cloud.example/api/v1/conformance'
  rel: http://www.opengis.net/def/rel/ogc/1.0/conformance
  type: application/json
  title: OGC Conformance Classes
- href: 'https://cloud.example/api/v1/openapi.json'
  rel: service-desc
  type: application/vnd.oai.openapi+json;version=3.0
  title: OpenAPI 3.0 description of the API
- href: 'https://cloud.example/api/v1/collections'
  rel: data
  type: application/json
  title: List of Datasets

4XX:
$ref: '#/components/responses/client_error'

5XX:
$ref: '#/components/responses/server_error'

/file_formats:
get:
  summary: Supported file formats
  operationId: list-file-types
  description: Lists supported input and output file formats.
  *Input* file formats specify which file a back-end can *read* from.
  *Output* file formats specify which file a back-end can *write* to.

The response to this request is an object listing all available input and output file formats separately with their parameters and additional data. This endpoint does not include the supported secondary web services.

**Note**: Format names and parameters MUST be fully aligned with the
GDAL codes if available, see [GDAL Raster Formats](https://gdal.org/drivers/raster/index.html) and [OGR Vector Formats](https://gdal.org/drivers/vector/index.html). It is OPTIONAL to support all output format parameters supported by GDAL. Some file formats not available through GDAL may be defined centrally for the geodatacube. Custom file formats or parameters MAY be defined.

The format descriptions must describe how the file formats relate to data cubes. Input file formats must describe how the files have to be structured to be transformed into data cubes. Output file formats must describe how the data cubes are stored at the back-end and how the resulting file structure looks like.

Back-ends MUST NOT support aliases, for example it is not allowed to support `geotiff` instead of `gtiff`. Nevertheless, geodatacube clients MAY translate user input input for convenience (e.g. translate `geotiff` to `gtiff`). Also, for a better user experience the back-end can specify a `title`.

Format names MUST be accepted in a case insensitive manner throughout the API.
format names that are used by clients and users, for example in process graphs.
additionalProperties:
  $ref: '#/components/schemas/file_format'
example:
  output:
    GTiff:
      title: GeoTiff
      description: Export to GeoTiff. Doesn't support cloud-optimized GeoTiffs (COGs) yet.
      gis_data_types:
        - raster
      parameters:
        tiled:
          type: boolean
          description: This option can be used to force creation of tiled TIFF files [true]. By default [false] stripped TIFF files are created.
          default: false
        compress:
          type: string
          description: Set the compression to use.
          default: NONE
          enum:
            - JPEG
            - LZW
            - DEFLATE
            - NONE
        jpeg_quality:
          type: integer
          description: Set the JPEG quality when using JPEG.
          minimum: 1
          maximum: 100
          default: 75
    links:
      - href: 'https://gdal.org/drivers/raster/gtiff.html'
        rel: about
        title: GDAL on the GeoTiff file format and storage options
GPKG:
  title: OGC GeoPackage
  gis_data_types:
    - raster
    - vector
  parameters:
    version:
      type: string
      description: Set GeoPackage version. In AUTO mode, this will be equivalent to 1.2 starting with GDAL 2.3.
      enum:
        - auto
        - '1'
        - '1.1'
        - '1.2'
      default: auto
  links:
      rel: about
      title: GDAL on GeoPackage for raster data
      rel: about
title: GDAL on GeoPackage for vector data

input:
GPKG:
title: OGC GeoPackage
gis_data_types:
- raster
- vector
parameters:
table:
type: string
description: >-
**RASTER ONLY.** Name of the table containing the
tiles. If the GeoPackage dataset only contains one
table, this option is not necessary. Otherwise, it
is required.

links:
  rel: about
title: GDAL on GeoPackage for raster data
  rel: about
title: GDAL on GeoPackage for vector data

4XX:
$ref: '#/components/responses/client_error'

5XX:
$ref: '#/components/responses/server_error'

/conformance:
get:
summary: Conformance classes this API implements
operationId: conformance
description: |
Lists all conformance classes specified in various standards that the
implementation conforms to. Conformance classes are commonly used in
all OGC APIs and the STAC API specification.

The conformance classes listed at this endpoint and listed in the
corresponding `conformsTo` property in `GET /` MUST be equal.

More details:
- [STAC API](https://github.com/radiantearth/stac-api-spec),
especially the conformance class "STAC API - Collections"
- [OGC APIs](https://ogcapi.ogc.org/)
tags:
- Capabilities
responses:
'200':
description: The URIs of all conformance classes supported by the
server.

content:
application/json:
schema:
title: OGC Conformance Classes
type: object
required:
- conformsTo
properties:
conformsTo:
  $ref: '#/components/schemas/conformsTo'

5XX:
$ref: '#/components/responses/server_error'

/collections:
get:
summary: Basic metadata for all datasets
operationId: list-collections
description: |-
Lists available collections with at least the required information.

It is **strongly RECOMMENDED** to keep the response size small by
omitting larger optional values from the objects in `collections` (e.
g. the
STAC `summaries` and `cube:dimensions` properties).
To get the full metadata for a collection clients MUST
request `GET /collections/{collection_id}`).

Note: Although it is possible to request public collections without
authorization, it is RECOMMENDED that clients (re-)request the
collections
with the Bearer token once available to also retrieve any private
collections.

**NOTE:** This endpoint may return collections from STAC API / openEO
API and OGC API - Coverages.
Distinguish them via the `stac_version` property which is always
present for STAC API / openEO API, but not for OGC API - Coverages.
tag:
- Data Discovery / Access
security:
- {}
- Bearer: []
parameters:
- $ref: '#/components/parameters/pagination_limit'
responses:
  '200':
    description: Lists of collections and related links.
    content:
      application/json:
        schema:
          title: Collections
          type: object
          required:
            - collections
            - links
          properties:
            collections:
              type: array
              items:
                allOf:
                - $ref: '#/components/schemas/collection'
                anyOf:
                - title: Coverage Collection
                - $ref: '#/components/schemas/stac_collection'
            links:
              $ref: '#/components/schemas/links_pagination'
example:
  collections:
  - stac_version: 1.0.0
    type: Collection
    id: Sentinel-2A
    title: Sentinel-2A MSI L1C
    description: Sentinel-2A is a wide-swath, high-resolution,
                multi-spectral imaging mission supporting Copernicus
                Land Monitoring studies, including the monitoring of
                vegetation, soil and water cover, as well as observation
                of inland waterways and coastal areas.
    license: proprietary
extent:
  spatial:
    bbox:
      - -180
      - 83
      - -90
      - 90
temporal:
  interval:
    - '2015-06-23T00:00:00Z'
    - '2019-01-01T00:00:00Z'

keywords:
- copernicus
- esa
- msi
- sentinel

providers:
- name: European Space Agency (ESA)
  roles:
  - producer
  - licensor
  url: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi

sentinel-2-msi
- name: Google Earth Engine
  roles:
  - host
  url: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2

links:
- rel: license

stac_version: 1.0.0
type: Collection
id: MOD09Q1
title: MODIS/Terra Surface Reflectance 8-Day L3 Global 250m SIN Grid V006
description: The MOD09Q1 Version 6 product provides an estimate of the surface spectral reflectance of Terra MODIS Bands 1-2 corrected for atmospheric conditions such as gasses, aerosols, and Rayleigh scattering. Provided along with the two 250 m MODIS bands is one additional layer, the Surface Reflectance QC 250 m band. For each pixel, a value is selected from all the acquisitions within the 8-day composite period. The criteria for the pixel choice include cloud and solar zenith. When several acquisitions meet the criteria the pixel with the minimum channel 3 (blue) value is used. Validation at stage 3 has been achieved for all MODIS Surface Reflectance products.

license: proprietary
temporal:
  interval:
    - '2000-02-01T00:00:00Z'
    - null
links:
  - rel: license
    href: 'https://geodatacube.example/api/v1/collections/MOD09Q1/license'
  - rel: alternate
    href: 'https://geodatacube.example/csw'
    title: OGC Catalogue Services 3.0
4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
'/collections/{collection_id}':
get:
  summary: Full metadata for a specific dataset
  operationId: describe-collection
  description: |
    Lists **all** information about a specific collection specified by the
    identifier `collection_id`.
    
    **Note:** Providing the Bearer token is REQUIRED for private collections.

    **NOTE:** This endpoint may return collections from STAC API / openEO
    API and OGC API - Coverages. Distinguish them via the `stac_version`
    property which is always present for STAC API / openEO API, but not for
    OGC API - Coverages.
  tags:
    - Data Discovery / Access
  security:
    - {}
    - Bearer: []
  parameters:
    - $ref: '#/components/parameters/collection_id'
  responses:
    '200':
      description: JSON object with the full collection metadata.
      content:
        application/json:
          schema:
            type: object
            allOf:
              - $ref: '#/components/schemas/collection'
            anyOf:
              - title: Coverage Collection
                required:
                  - cube:dimensions
                summaries
                allOf:
                  - $ref: '#/components/schemas/stac_collection'
              - $ref: '#/components/schemas/stac_collection'
            example:
              stac_version: 1.0.0
              stac_extensions:
                - https://stac-extensions.github.io/datacube/v2.2.0/schema.

            json
            type: Collection
            id: Sentinel-2
            title: Sentinel-2 MSI L2A
            description: >-
              Sentinel-2A is a wide-swath, high-resolution, multi-spectral
imaging mission supporting Copernicus Land Monitoring studies.

license: proprietary

keywords:
- copernicus
- esa
- msi
- sentinel

providers:
- name: European Space Agency (ESA)
  roles:
  - producer
  - licensor
  url: >-
    https://sentinel.esa.int/web/sentinel/user-guides/
sentinel-2-msi
- name: Google
  roles:
  - host
  url: >-
    https://developers.google.com/earth-engine/datasets/
catalog/COPERNICUS_S2
  extent:
  spatial:
    bbox:
    - -180
    - -56
    - 180
    - 83
  temporal:
    interval:
    - '2015-06-23T00:00:00Z'
    - null
  links:
  - rel: license
    type: application/pdf
  - rel: http://www.opengis.net/def/rel/ogc/1.0/queryables
    href: https://geodatacube.example/api/v1/collections/Sentinel-2A/queryables
    type: application/schema+json
  - rel: about
    href: https://earth.esa.int/web/sentinel/user-guides/
sentinel-2-msi/product-types/level-1c
    type: text/html
    title: ESA Sentinel-2 MSI Level-1C User Guide
  - rel: example
    href: 'https://geodatacube.example/api/v1/collections/Sentinel-2/examples/true-color.json'
    type: application/json
    title: Example Process for True-Color Visualization
  - rel: example
    href: 'https://geodatacube.example/api/v1/collections/Sentinel-2/examples/ndvi.json'
    type: application/json
    title: Example Process for NDVI Calculation and Visualization

'cube:dimensions':
  x:
    type: spatial
    axis: x
    extent:
- -180
- 180

reference_system: 4326

y:
  type: spatial
  axis: 'y'
  extent:
    -56
    83

reference_system: 4326

t:
  type: temporal
  extent:
    '2015-06-23T00:00:00Z'
    null
  step: null

bands:
  type: bands
  values:
    - B1
    - B2
    - B3
    - B4
    - B5
    - B6
    - B7
    - B8
    - B8A
    - B9
    - B10
    - B11
    - B12

summaries:
  'constellation':
    - Sentinel-2
  'platform':
    - Sentinel-2A
    - Sentinel-2B
  'instruments':
    - MSI
  'eo:cloud_cover':
    minimum: 0
    maximum: 75
  'sat:orbit_state':
    - ascending
    - descending
  'gsd':
    - 10
    - 20
    - 60
  'eo:bands':
    - name: B1
      common_name: coastal
      center_wavelength: 0.4439
      gsd: 60
    - name: B2
      common_name: blue
      center_wavelength: 0.4966
      gsd: 10
    - name: B3
      common_name: green
      center_wavelength: 0.56
      gsd: 10
- name: B4
c  common_name: red
c  center_wavelength: 0.6645
gsd: 10
- name: B5
c  center_wavelength: 0.7039
gsd: 20
- name: B6
c  center_wavelength: 0.7402
gsd: 20
- name: B7
c  center_wavelength: 0.7825
gsd: 20
- name: B8
c  common_name: nir
c  center_wavelength: 0.8351
gsd: 10
- name: B8A
c  common_name: nir08
c  center_wavelength: 0.8648
gsd: 20
- name: B9
c  common_name: nir09
c  center_wavelength: 0.945
gsd: 60
- name: B10
c  common_name: cirrus
c  center_wavelength: 1.3735
gsd: 60
- name: B11
c  common_name: swir16
c  center_wavelength: 1.6137
gsd: 20
- name: B12
c  common_name: swir22
c  center_wavelength: 2.2024
gsd: 20

'proj:epsg':
  minimum: 32601
  maximum: 32660
assets:
thumbnail:
  href: 'https://geodatacube.example/api/v1/collections/Sentinel-2/thumbnail.png'
type: image/png
  title: Preview
  roles:
  - thumbnail
inspire:
  href: 'https://geodatacube.example/api/v1/collections/Sentinel-2/inspire.xml'
type: application/xml
  title: INSPIRE metadata
  description: INSPIRE compliant XML metadata
  roles:
  - metadata

4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
'/collections/{collection_id}/queryables':
get:
  summary: Metadata filters for a specific dataset
operationId: list-collection-queryables

description: |-
Lists **all** supported metadata filters (also called "queryables") for a specific collection.

This endpoint is compatible with endpoint defined in the STAC API extension
[`filter`](https://github.com/stac-api-extensions/filter#queryables) and
For a precise definition please follow those specifications.

This endpoints provides a JSON Schema for each queryable that geodatacube users can use in multiple scenarios:
1. For loading data from the collection, e.g. in the process `load_collection`
2. For filtering items using CQL2 on the `/collections/{collection_id}/items` endpoint.

Note: Providing the Bearer token is REQUIRED for private collections.

tags:
- Data Discovery / Access
- OGC API - Features / STAC API

security:
- {}
- Bearer: []

parameters:
- $ref: '#/components/parameters/collection_id'

responses:

200:
  description: |-
  A JSON Schema defining the queryables.
  It is RECOMMENDED to dereference all "$refs".
  
  content:
  application/schema+json:
    schema:
      $ref: '#/components/schemas/json_schema'
    example:
      $schema: https://json-schema.org/draft/2019-09/schema
      $id: https://geodatacube.example/api/v1/collections/Sentinel-2A/queryables

type: object
  title: Sentinel 2A
  properties:
    'eo:cloud_cover':
      title: Cloud Cover
      type: number
      minimum: 0
      maximum: 100
    platform:
      title: Platform
      description: The satellite platform.
      type: string
      enum:
        - sentinel-2a
        - sentinel-2b
    additionalProperties: false

4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
/collections/{collection_id}/items:
  get:
    tags:
      - OGC API - Features / STAC API
    summary: Fetch Features / Items
    description: Fetch features of the feature collection with id `collection_id`.
    operationId: list-items
    parameters:
      - $ref: '#/components/parameters/collection_id'
      - $ref: '#/components/parameters/pagination_limit'
      - $ref: '#/components/parameters/bbox'
      - $ref: '#/components/parameters/datetime'
    security:
      - {}
    responses:
      "200":
        description: The response is a document consisting of features in the collection.
        content:
          application/geo+json:
            schema:
              allOf:
                - $ref: '#/components/schemas/GeoJsonFeatureCollection'
                - type: object
                  required:
                    - features
                  properties:
                    features:
                      type: array
                      items:
                        $ref: '#/components/schemas/stac_item'
get:
  tags:
  - OGC API - Features / STAC API
  summary: Fetch a Feature / Item
  description: |
    Fetch the feature with id `feature_id` in the feature collection with id `collection_id`.
  operationId: describe-item
  security:
  - Bearer: []
  parameters:
  - $ref: '#/components/parameters/collection_id'
  - $ref: '#/components/parameters/feature_id'
  responses:
    "200":
      description: |
        Fetch the feature with id `feature_id` in the feature collection with id `collection_id`.
      content:
        application/geo+json:
          schema:
            allOf:
              - $ref: '#/components/schemas/GeoJsonFeature'
              - $ref: '#/components/schemas/stac_item'
    4XX:
      $ref: '#/components/responses/client_error_auth'
    5XX:
      $ref: '#/components/responses/server_error'

get:
  tags:
  - Data Discovery / Access
  - OGC API - Coverages
  summary: Retrieve a coverage
  description: |
    Coverage identified by {collection_id}.
    Use content negotiation to request required format.
  operationId: describe-coverage
  security:
  - Bearer: []
  parameters:
  - $ref: '#/components/parameters/collection_id'
  - $ref: '#/components/parameters/subset'
  - $ref: '#/components/parameters/bbox'
  - $ref: '#/components/parameters/datetime'
  - $ref: '#/components/parameters/properties'
  - $ref: '#/components/parameters/scale-factor'
  - $ref: '#/components/parameters/scale-axes'
- "$ref": 
  
- "$ref": 
  
- "$ref": 
  
- "$ref": 
  
- "$ref": 

responses:
'200':
  description: A full coverage.
  content:
    application/json:
      schema:
        "$ref": 
    image/tiff; application=geotiff:
      schema:
        type: string
        format: binary
    multipart/related:
      schema:
        type: string
        format: binary
    text/html:
      schema:
        type: string
  4XX:
    $ref: 
  5XX:
    $ref: 
"/collections/{collection_id}/coverage/domainset":
get:
  tags:
  - Data Discovery / Access
  - OGC API - Coverages
  summary: Retrieve a coverage's domainset
  description: a coverage's domainset; use content negotiation to request HTML or JSON
  operationId: describe-coverage-domainset
  security:
  - 
    - Bearer: []
  parameters:
  - "$ref": 
  - "$ref": 
  - "$ref": 
  - "$ref": 
  - "$ref": 
  - "$ref": 
  - "$ref": 
    "$ref": 
  responses:
    '200':
      description: A coverages domainset.
      content:
        application/json:
          schema:
            "$ref": 
        text/html:
          schema:
            type: string
      4XX:
        $ref: 
      5XX:
        $ref: 
"/collections/{collection_id}/coverage/rangetype":
OPEN GEOSPATIAL CONSORTIUM 23-048
38
get:
  tags:
  - Data Discovery / Access
  - OGC API - Coverages
  summary: Retrieve a coverage's rangetype
  description: a coverage's rangetype; use content negotiation to request HTML or JSON
  operationId: describe-coverage-rangetype
  security:
  - {}
  - Bearer: []
  parameters:
  - $ref: "#/components/parameters/collection_id"
  - $ref: "#/components/parameters/f-rangetype"
  responses:
    '200':
      description: A coverage's rangetype.
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/rangeType"
        text/html:
          schema:
            type: string
    4XX:
      $ref: "#/components/responses/client_error_auth"
    5XX:
      $ref: "#/components/responses/server_error"

/processes:
  get:
  summary: Supported predefined processes
  operationId: list-processes
  description: |
    Lists all predefined processes and returns detailed process descriptions, including parameters and return values.
    
    **NOTE:** This endpoint may return processes from openEO and OGC API - Processes. Distinguish them via the `version` property (OGC API) and the `parameters` / `returns` (openEO) properties.
  tags:
  - Process Discovery
  - OGC API - Processes
  - openEO
  security:
  - {}
  - Bearer: []
  parameters:
  - $ref: "#/components/parameters/pagination_limit"
  responses:
    '200':
      description: Formal specification describing the supported predefined processes.
      content:
        application/json:
          schema:
            title: Processes
            type: object
            required:
              - processes
              - links
            properties:
              processes:
type: array
items:
oneOf:
  - title: openEO Predefined Process
description: A predefined process made available by the back-end.
type: object
required:
  - id
  - description
  - parameters
  - returns
allOf:
  - $ref: '#/components/schemas/process'
  - title: OGC API Process
allOf:
  - $ref: '#/components/schemas/ogc_processSummary'
links:
  $ref: '#/components/schemas/links_pagination'
example:
  processes:
  - id: apply
    summary: Apply a process to each pixel
    description: |
      Applies a *unary* process to each pixel value in the data cube (i.e. a local operation). A unary process takes a single value and returns a single value, for example \`abs()\` or \`linear_scale_range()\``.
categories:
  - cubes
parameters:
  - name: data
description: A data cube.
schema:
  type: object
  subtype: datacube
  - name: process
description: 'A unary process to be applied on each value, may consist of multiple sub-processes.'
schema:
  type: object
  subtype: process-graph
parameters:
  - name: x
description: The value to process.
schema:
  description: Any data type.
returns:
description: 'A data cube with the newly computed values. The resolution, cardinality and the number of dimensions are the same as for the original data cube.'
schema:
  type: object
  subtype: datacube
- id: multiply
  summary: Multiplication of two numbers
description: |
    Multiplies the two numbers `x` and `y` (*x * y*) and returns the computed product.
    No-data values are taken into account so that `null` is returned if any element is such a value.
The computations follow [IEEE Standard 754](https://ieeexplore.ieee.org/document/8766229) whenever the processing environment supports it.

categories:
- math

parameters:
- name: x
description: The multiplier.
schema:
  type:
  - number
  - 'null'
- name: 'y'
description: The multiplicand.
schema:
  type:
  - number
  - 'null'

returns:
description: The computed product of the two numbers.
schema:
  type:
  - number
  - 'null'

exceptions:
  MultiplicandMissing:
    message: Multiplication requires at least two numbers.

examples:
- arguments:
  x: 5
  y: 2.5
returns: 12.5
- arguments:
  x: -2
  y: -4
returns: 8
- arguments:
  x: 1
  y: null
returns: null

links:
- rel: about
  href: 'http://mathworld.wolfram.com/Product.html'
title: Product explained by Wolfram MathWorld
- rel: about
title: IEEE Standard 754-2019 for Floating-Point

Arithmetic

links:
- rel: alternate
  href: 'https://geodatacube.example/processes'
type: text/html
title: HTML version of the processes

/processes/{processID}:
  get:
    tags:
    - Process Discovery
    - OGC API - Processes
    summary: Retrieve an OGC API process description
description: |
  The process description contains information about inputs and outputs and a link to the execution-endpoint for the process. The Core does not mandate the use of a specific process description to specify the interface
of a process. That said, the Core requirements class makes the following recommendation:

Implementations SHOULD consider supporting the OGC process description.

For more information, see [Section 7.10](https://docs.ogc.org/is/18-062/18-062.html#sc_process_description).

```json
operationId: describe-ogc-process
security:
  - {}  
  - Bearer: []
parameters:
  - $ref: '#/components/parameters/ogc_processID'
responses:
  "200":
    description: A process description.
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ogc_process'
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'
```

/processes/{processID}/execution:

```json
post:
  tags:
    - OGC API - Processes
  summary: OGC API / Execute a process
  description: | Create a new job.
  operationId: execute-ogc-process
  security:
    - Bearer: []
  parameters:
    - $ref: '#/components/parameters/ogc_processID'
  requestBody:
    description: Mandatory execute request JSON
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ogc_execute'
        required: true
  responses:
    "200":
      description: Result of synchronous execution
      content:
        */:
          schema:
            description: Any kind of data could be returned.
    "201":
      description: Started asynchronous execution. Created job.
      headers:
        Location:
          description: URL to check the status of the execution/job.
          style: simple
          explode: false
          schema:
            type: string
```
The preference applied to execute the process asynchronously (see RFC 2740).

```json

description: The preference applied to execute the process asynchronously (see. RFC 2740).
style: simple
explode: false
schema:
  type: string
content:
  application/json:
    schema:
      $ref: '#/components/schemas/ogc_statusInfo'
4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
callbacks:
  jobCompleted:
    '{request.body#/subscriber/successUri}':
      post:
        requestBody:
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ogc_results'
responses:
  "200":
    description: Results received successfully

/credentials/basic:
get:
  summary: HTTP Basic authentication
  operationId: authenticate-basic
  description: Checks the credentials provided through [HTTP Basic Authentication according to RFC 7617](https://www.rfc-editor.org/rfc/rfc7617.html) and returns an access token for valid credentials.

The credentials (username and password) MUST be sent in the HTTP header `Authorization` with type `Basic` and the Base64 encoded string consisting of username and password separated by a double colon `:`. The header would look as follows for username `user` and password `pw`: `Authorization: Basic dXNlcjpwdw==`.

The access token has to be used in the Bearer token for authorization in subsequent API calls (see also the information about Bearer tokens in this document). The access token returned by this request MUST NOT be provided with `basic//` prefix, but the Bearer Token sent in subsequent API calls to protected endpoints MUST be prefixed with `basic//`. The header in subsequent API calls would look as follows: `Authorization: Bearer basic//TOKEN` (replace `TOKEN` with the actual access token).

It is RECOMMENDED to implement this authentication method for non-public services only.

tags:
- Account Management
security:
- Basic: []
responses:
'200':
  description: Credentials are correct and authentication succeeded.
  content:
    application/json:
      schema:
        title: HTTP Basic Access Token
        type: object
        required:
          - access_token
        properties:
          access_token:
            description: The access token (without `basic//` prefix) to be used in the Bearer token for authorization in subsequent API calls.
            type: string
            example: b34ba2bdf9ac9ee1

4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
  $ref: '#/components/responses/server_error'

/credentials/oidc:
  get:
    summary: OpenID Connect authentication
    operationId: authenticate-oidc
    description: |-
      Lists the supported [OpenID Connect](http://openid.net/connect/) providers (OP). OpenID Connect Providers MUST support [OpenID Connect Discovery](http://openid.net/specs/openid-connect-discovery-1_0.html). It is highly RECOMMENDED to implement OpenID Connect for public services in favor of Basic authentication.

      GGC clients MUST use the **access token** as part of the Bearer token for authorization in subsequent API calls (see also the information about Bearer tokens in this document). Clients MUST NOT use the id token or the authorization code. The access token provided by an OpenID Connect provider does not necessarily provide information about the issuer (i.e. the OpenID Connect provider) and therefore a prefix MUST be added to the Bearer Token sent in subsequent API calls to protected endpoints. The Bearer Token sent to protected endpoints MUST consist of the authentication method (here `oidc`), the provider ID and the access token itself. All separated by a forward slash `/`. The provider ID corresponds to the value specified for `id` for each provider in the response body of this endpoint. The header in subsequent API calls for a provider with `id` `ms` would look as follows: `Authorization: Bearer oidc/ms/TOKEN` (replace `TOKEN` with the actual access token received from the OpenID Connect Provider).

      Back-ends MAY request user information ([including Claims](https://openid.net/specs/openid-connect-core-1_0.html#Claims)) from the [OpenID Connect Userinfo endpoint](https://openid.net/specs/openid-connect-core-1_0.html#UserInfo) using the access token (without the prefix described above). Therefore, both openEO client and openEO back-end are relying parties (clients) to the OpenID Connect Provider.

    tags:
      - Account Management
security:
- {}
responses:
  '200':
    description: Lists the OpenID Connect Providers.
    content:
      application/json:
        schema:
          title: OpenID Connect Providers
          type: object
          required:
          - providers
          properties:
            providers:
              type: array
              description: The first provider in this list is the default provider for authentication.
              minItems: 1
              items:
                title: OpenID Connect Provider
                type: object
                required:
                - id
                - issuer
                - title
                properties:
                  id:
                    type: string
                    description: A per-backend **unique** identifier for the OpenID Connect Provider to be as prefix for the Bearer token.
                    pattern: '^[\d\w]{1,20}$'
                  issuer:
                    type: string
                    format: uri
                    description: The [issuer location](https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig) (also referred to as 'authority' in some client libraries) is the URL of the OpenID Connect provider, which conforms to a set of rules:
          
          1. After appending `\./well-known/openid-configuration` to the URL, a [HTTP/1.1 GET request](https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfigurationRequest) to the concatenated URL MUST return a [OpenID Connect Discovery Configuration Response](https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfigurationResponse). The response provides all information required to authenticate using OpenID Connect.
          2. The URL MUST NOT contain a terminating forward slash `\/`. 
example: 'https://accounts.google.com'
scopes:
  type: array
  description: >-
    A list of OpenID Connect scopes that the client
    MUST at least include when requesting authorization.
    Clients MAY add additional scopes such as the
    `offline_access` scope to retrieve a refresh token.
    If scopes are specified, the list MUST at least
    contain the `openid` scope.
  items:
    type: string
title:
  type: string
  description: >-
    The name that is publicly shown in clients for this
    OpenID Connect provider.
description:
  type: string
  format: commonmark
  description: |-
    A description that explains how the authentication
    procedure works.
credentials. This should
  It should make clear how to register and get
  include instruction on setting up `client_id`,
  `client_secret` and `redirect_uri`.
MAY be used for rich
  [CommonMark 0.29](http://commonmark.org/) syntax
text representation.
default_clients:
  title: Default OpenID Connect Clients
  type: array
  description: |-
    List of default OpenID Connect clients that can be
    used by an openEO client for OpenID Connect based authentication.
back-end implementer. A default OpenID Connect client is managed by the
client secret, It MUST be configured to be usable without a
grant types like
  "Authorization Code Grant with PKCE" and "Device
Authorization Grant with PKCE"
without availability guarantees.
A default OpenID Connect client is provided
update it any time. The back-end implementer CAN revoke, reset or
default OpenID Connect client information
  As such, openEO clients SHOULD NOT store or cache
for long term usage.
A default OpenID Connect client is intended to
  simplify authentication for novice users.
simplify authentication for
For production use cases, it is RECOMMENDED to set
up a dedicated OpenID Connect client.
uniqueItems: true
items:
  title: Default OpenID Connect Client
  type: object
  required:
    - id
    - grant_types
  properties:
    id:
      type: string
      description: The OpenID Connect Client ID to be used in the authentication procedure.
    grant_types:
      type: array
      description: List of authorization grant types (flows) supported by the OpenID Connect client.
      A grant type descriptor consist of a OAuth 2.0 grant type, with an additional `+pkce` suffix when the grant type should be used with the PKCE extension as defined in [RFC 7636](https://www.rfc-editor.org/rfc/rfc7636.html).
      Allowed values:
      - `implicit`: Implicit Grant as specified in [RFC 6749, sec. 1.3.2](https://www.rfc-editor.org/rfc/rfc6749.html#section-1.3.2)
      - `authorization_code` / `authorization_code+pkce`: Authorization Code Grant as specified in [RFC 6749, sec. 1.3.1](https://www.rfc-editor.org/rfc/rfc6749.html#section-1.3.1), with or without PKCE extension.
      - `urn:ietf:params:oauth:grant-type:device_code` / `urn:ietf:params:oauth:grant-type:device_code+pkce`: Device Authorization Grant (aka Device Code Flow) as specified in [RFC 8628](https://www.rfc-editor.org/rfc/rfc8628.html), with or without PKCE extension. Note that the combination of this grant with the PKCE extension is not standardized yet.
      - `refresh_token`: Refresh Token as specified in [RFC 6749, sec. 1.5](https://www.rfc-editor.org/rfc/rfc6749.html#section-1.5)

minItems: 1
uniqueItems: true
items:
  type: string
  enum:
    - 'implicit'
    - 'authorization_code'
    - 'authorization_code+pkce'
    - 'urn:ietf:params:oauth:grant-type:device_code'
    - 'urn:ietf:params:oauth:grant-type:device_code+pkce'
    - 'refresh_token'

redirect_urls:
  type: array
  description: List of redirect URLs that are whitelisted by the OpenID Connect client. Redirect URLs MUST be provided when the OpenID Connect client supports the Implicit Grant or the Authorization Code Grant (with or without PKCE extension).
uniqueItems: true
items:
  type: string
  format: uri

links:
  type: array
  description: |
  Links related to this provider, for example a
  help page or a page to register a new user account.

For relation types see the lists of
[common relation types in openEO](#section/API-Principles/Web-Linking).

example:
  $ref: '#/components/schemas/link'

providers:
- id: egi
  issuer: 'https://aai.egi.eu/oidc'
  title: EGI (default)
  description: Login with your academic account.
  scopes:
  - openid
  - profile
  - email
  default_clients:
  - id: KStcUzD5AIUA
    grant_types:
    - implicit
    - authorization_code+pkce
    - urn:ietf:params:oauth:grant-type:device_code+pkce
    - refresh_token
    redirect_urls:
    - https://editor.openeo.org/
- id: google
  issuer: 'https://accounts.google.com'
  title: Google
  description: Login with your Google Account.
  scopes:
  - openid
  - profile
  - email
  - earthengine
- id: ms
  issuer: 'https://login.microsoftonline.com/example-tenant/v2.0'
  title: Microsoft
  description: Login with your Microsoft or Skype Account.
  scopes: []

4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'

/post:
  summary: Process and download data synchronously
  operationId: compute-result
  description: >-
  Executes a user-defined process directly (synchronously) and the
  result will be downloaded in the format specified in the process graph. This endpoint
  can be used to generate small previews or test user-defined processes
  before
starting a batch job.

Timeouts on either client- or server-side are to be expected for complex computations.
Back-ends MAY send the an error immediately if the computation is expected to time out.
Otherwise requests MAY time-out after a certain amount of time by sending an error.

tags:
- openEO
security:
- Bearer: []
responses:
  '200':
    description: Result data in the requested output format
    headers:
      Content-Type:
        description: The appropriate media (MIME) type for the requested output format MUST be sent, if the response contains a single file.
        To mimic the results of batch jobs, it is RECOMMENDED that
        1. clients extract the tar file directly after receiving it so that users can directly work on the contained files *and*
        2. back-ends add STAC Items and/or Collections to the tar file so that users can make sense of the files.
    schema:
      type: string
    Link:
      description: The header MAY indicate a link to a log file generated by the request. If provided, the link MUST be serialized according to [RFC 8288](https://www.rfc-editor.org/rfc/rfc8288.html#section-3) and MUST use the relation type `monitor`. The link MUST follow the specifications for the links `GET /jobs/{job_id}/logs` and `GET /services/{service_id}/logs`, except that is MUST NOT accept any parameters (limit/offset). Therefore, the link MUST be accessible with HTTP GET, MUST be secured using a Bearer token and MUST follow the corresponding request body schema.
      schema:
        type: string
        pattern: ^<[^>]+>;s?rel="monitor"
        example: <https://geodatacube.example/api/v1/logs/258489231>;

4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
requestBody:
  description: 'Specifies the job details, e.g. the user-defined process and billing details.'
  required: true
  content:
    application/json:
      schema:
        title: Synchronous Result Request
        type: object
        required:
### /process_graphs

**get:**

**summary:** List all user-defined openEO processes

**operationId:** list-custom-processes

**description:**

Lists all user-defined processes (process graphs) of the authenticated user that are stored at the back-end.

It is **strongly RECOMMENDED** to keep the response size small by omitting larger optional values from the objects in `processes` (e.g. the `exceptions`, `examples` and `links` properties).

To get the full metadata for a user-defined process clients MUST request `GET /process_graphs/{process_graph_id}`.

**tags:**
- Process Discovery
- openEO - User-Defined Processes

**security:**
- Bearer: []

**parameters:**
- $ref: '#/components/parameters/pagination_limit'

**responses:**

**'200':**

**description:** JSON array with user-defined processes.

**content:**

application/json:

**schema:**

- **title:** User-Defined Processes
- **type:** object
- **required:**
  - **processes**
  - **links**

**processes:**

- **description:** Array of user-defined processes
- **type:** array
- **items:**
  - $ref: '#/components/schemas/user_defined_process_meta'

**links:**

- $ref: '#/components/schemas/links_pagination'

**example:**

- **id:** evi
- **summary:** Enhanced Vegetation Index
- **description:** Computes the Enhanced Vegetation Index (EVI).
  - **type:** number

**parameters:**

- **name:** red
  - **description:** Value from the red band.
  - **schema:**
    - **type:** number

- **name:** blue
  - **description:** Value from the blue band.
schema:
  type: number
- name: nir
description: Value from the near infrared band.
schema:
  type: number
returns:
description: Computed EVI.
schema:
  type: number
- id: ndsi
summary: Normalized-Difference Snow Index
parameters:
- name: green
description: Value from the Visible Green (0.53 - 0.61 micrometers) band.
schema:
  type: number
- name: swir
description: Value from the Short Wave Infrared (1.55 - 1.75 micrometers) band.
schema:
  type: number
returns:
schema:
  type: number
- id: my_custom_process
links: []
4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
/process_graphs/{process_graph_id}:
parameters:
- name: process_graph_id
  in: path
description: Per-user unique identifier for a user-defined process.
required: true
schema:
  $ref: '#/components/schemas/process_id'
get:
  summary: Full metadata for a user-defined process
  operationId: describe-custom-process
description: Lists all information about a user-defined process,
  including its process graph.
tag:
  - openEO - User-Defined Processes
security:
  - Bearer: []
responses:
  '200':
description: The user-defined process with process graph.
content:
  application/json:
schema:
    title: User-Defined Process
description: A user-defined process with processing
instructions as process graph.
type: object
required:
- process_graph
allOf:
- $ref: '#/components/schemas/user_defined_process_meta'
examples:
  evi_user_defined_process:
    $ref: '#/components/examples/evi_user_defined_process'

4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'

put:
  summary: Store a user-defined process
  operationId: store-custom-process
  description: |
    Stores a provided user-defined process with process graph that can be
    reused in other processes.
    
    If a process with the specified `process_graph_id` exists, the process
    is fully replaced. The id can't be changed for existing user-defined
    processes. The id MUST be unique across its namespace.
    
    Partially updating user-defined processes is not supported.
    
    To simplify exchanging user-defined processes, the property `id` can
    be part of
    the request body. If the values don't match, the value for `id` gets
    replaced with the value from the `process_graph_id` parameter in the
    path.
  tags:
    - openEO - User-Defined Processes
  security:
    - Bearer: []
  responses:
    '200':
      description: The user-defined process has been stored successfully.
    4XX:
      $ref: '#/components/responses/client_error_auth'
    5XX:
      $ref: '#/components/responses/server_error'
  requestBody:
    required: true
    description: Specifies the process graph with its meta data.
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/process_graph_with_metadata'
        examples:
          evi_user_defined_process:
            $ref: '#/components/examples/evi_user_defined_process'

delete:
  summary: Delete a user-defined process
  operationId: delete-custom-process
  description: |
    Deletes the data related to this user-defined process, including its
    process graph.
    
    Does NOT delete jobs or services that reference this user-defined
    process.
  tags:
    - openEO - User-Defined Processes
  security:
    - Bearer: []
  responses:
    '204':
      description: The user-defined process has been successfully deleted
$ref: '#/components/responses/client_error_auth'

5XX:
$ref: '#/components/responses/server_error'

/service_types:
  get:
    summary: Supported secondary web service protocols
    operationId: list-service-types
    description: Lists supported secondary web service protocols such as [OGC WMS](http://www.opengeospatial.org/standards/wms), [OGC WCS](http://www.opengeospatial.org/standards/wcs), [OGC API - Features](https://www.ogc.org/standards/ogcapi-features) or [XYZ tiles](https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames).
    The response is an object of all available secondary web service protocols with their supported configuration settings and expected process parameters.

    * The configuration settings for the service SHOULD be defined upon creation of a service and the service will be set up accordingly.
    * The process parameters SHOULD be referenced (with a `from_parameter` reference) in the user-defined process that is used to compute web service results.

      The appropriate arguments MUST be provided to the user-defined process, usually at runtime from the context of the web service, For example, a map service such as a WMS would need to inject the spatial extent into the user-defined process so that the back-end can compute the corresponding tile correctly.

      To improve interoperability between back-ends common names for the services SHOULD be used, e.g. the abbreviations used in the official [OGC Schema Repository](http://schemas.opengis.net/) for the respective services.

      Service names MUST be accepted in a *case insensitive* manner throughout the API.

    tags: 
      - openEO - Secondary Services (OGC APIs)
    security:
      - {} 
      - Bearer: []
    responses:
      '200':
        description: An object with a map containing all service names as keys and an object that defines supported configuration settings and process parameters.
        content:
          application/json:
            schema:
              title: Service Types
              type: object
              description: Map of supported secondary web services.
              additionalProperties:
                x-additionalPropertiesName: Service Name
                title: Service Type
                type: object
                required:
                  - configuration
- process_parameters
  properties:
    title:
      $ref: '#/components/schemas/object_title'
    description:
      $ref: '#/components/schemas/description'
    deprecated:
      $ref: '#/components/schemas/deprecated'
    experimental:
      $ref: '#/components/schemas/experimental'
  configuration:
    title: Service Configuration
    description: Map of supported configuration settings
made available to the creator of the service.

  type: object
  additionalProperties:
    $ref: '#/components/schemas/resource_parameter'

process_parameters:
  title: Process Parameters
  description: List of parameters made available to user-defined processes.

  type: array
  items:
    $ref: '#/components/schemas/process_parameter'

links:
  description: |
    Links related to this service type, e.g. more information about the configuration settings and
process parameters.

For relation types see the lists of
[common relation types](#section/API-Principles/Web-Linking).

  type: array
  items:
    $ref: '#/components/schemas/link'

example:
  WMS:
    title: OGC Web Map Service
    configuration:
      version:
        type: string
        description: The WMS version offered to consumers of the service.
        default: 1.3.0
        enum:
          - 1.1.1
          - 1.3.0
    process_parameters:
      - name: layer
        description: The layer name.
        schema:
          type: string
          default: roads
      - name: spatial_extent
        description: A bounding box in WGS84.
        schema:
          type: object
          required:
            - west
            - south
            - east
            - north
properties:
  west:
    description: West (lower left corner, coordinate axis 1).
    type: number
  south:
    description: South (lower left corner, coordinate axis 2).
    type: number
  east:
    description: East (upper right corner, coordinate axis 1).
    type: number
  north:
    description: North (upper right corner, coordinate axis 2).
    type: number

links:
  - href: 'https://www.opengeospatial.org/standards/wms'
    rel: about
    title: OGC Web Map Service Standard

OGC-API-FEATURES:
  title: OGC API - Features
  description: Exposes an OGC API - Features in version 1.0 of the specification (successor of OGC WFS 3.0).

configuration:
  title:
    type: string
  description:
    type: string
  landing page

conformsTo:
  type: array
  description:
    The OGC API - Features conformance classes to enable for this service.
  items:
    type: string
    enum:
      - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core
      - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30
      - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/html
      - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/geojson
      - http://www.opengis.net/spec/ogcapi-features-2/1.0/conf/crs

process_parameters: []

links:
  - href: 'https://www.opengeospatial.org/standards/wfs'
    rel: about
    title: OGC Web Feature Service Standard

4XX: $ref: '#/components/responses/client_error'
5XX: $ref: '#/components/responses/server_error'

/services:
get:
  summary: List all web services
  operationId: list-services
  description: |
    Lists all secondary web services submitted by a user.

    It is **strongly** RECOMMENDED** to keep the response size small by
    omitting
    all optional non-scalar values (i.e. arrays and objects) from objects
    in `services`
    (i.e. the `process`, `configuration` and `attributes` properties).
    To get the full metadata for a secondary web service clients MUST
    request `GET /services/{service_id}`.
  tags:
  - openEO - Secondary Services (OGC APIs)
  security:
  - Bearer: []
  parameters:
  - $ref: '#/components/parameters/pagination_limit'
  responses:
    '200':
      description: Array of secondary web service descriptions
      content:
        application/json:
          schema:
            title: Secondary Web Services
            type: object
            required:
              - services
              - links
            properties:
              services:
                type: array
                items:
                  $ref: '#/components/schemas/service'
              links:
                $ref: '#/components/schemas/links_pagination'
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'

post:
  summary: Publish a new service
  operationId: create-service
  description: |
    Creates a new secondary web service such as a
    [OGC WMS](http://www.opengeospatial.org/standards/wms),
    [OGC WCS](http://www.opengeospatial.org/standards/wcs),
    [OGC API - Features](https://www.ogc.org/standards/ogcapi-features)
    or [XYZ tiles](https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames).

    The secondary web service SHOULD process the underlying
data on demand, based on process parameters provided to the
user-defined process (through `from_parameter` references) at run-time,
for example for the spatial/temporal extent, resolution, etc.
The available process parameters are specified per
service type at `GET /service_types`.

    **Note:** Costs incurred by shared secondary web services are usually
paid by the owner, but this depends on the service type and whether it
supports charging fees or not.
  tags:
- openEO - Secondary Services (OGC APIs)

security:
  - Bearer: []

responses:
  '201':
    description: The service has been created successfully.
    headers:
      Location:
        required: true
        schema:
          description: Absolute URL to the newly created service.
          The URL points to the metadata endpoint `GET /services/{service_id}` with the `{service_id}` being
          the unique identifier (ID) of the created service.
          MUST NOT point to the actual instance (e.g. WMTS base URL) of
          the service. The URL to the instance is made available by the
          metadata endpoint in the property `url`.
          format: uri
          type: string
          example: 'https://geodatacube.example/api/v1/services/123'
    GDC-Identifier:
      required: true
      schema:
        $ref: '#/components/schemas/service_id'
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'
requestBody:
  required: true
  content:
    application/json:
      schema:
        title: Store Secondary Web Service Request
        type: object
        required:
          - type
            - process
        properties:
          title:
            $ref: '#/components/schemas/eo_title'
          description:
            $ref: '#/components/schemas/eo_description'
          process:
            $ref: '#/components/schemas/process_graph_with_metadata'
          type:
            $ref: '#/components/schemas/service_type'
          enabled:
            allOf:
              - $ref: '#/components/schemas/service_enabled'
              - default: true
          configuration:
            $ref: '#/components/schemas/service_configuration'
          log_level:
            $ref: '#/components/schemas/min_log_level_default'
        description: You can add additional back-end specific
        properties here.
        description: The base data for the secondary web service to create
        `/services/{service_id}`:
parameters:
  - $ref: '#/components/parameters/service_id'

patch:
  summary: Modify a service
  operationId: update-service
  description: |
    Modifies an existing secondary web service at the back-end,
    but maintain the identifier. Changes can be grouped in a single request.
    
    User have to create a new service to change the service type.

tags:
  - openEO - Secondary Services (OGC APIs)

security:
  - Bearer: []

responses:
  '204':
    description: Changes to the service were applied successfully.
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'

requestBody:
  required: true
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/UpdateSecondaryWebServiceRequest'

get:
  summary: Full metadata for a service
  operationId: describe-service
  description: Lists all information about a secondary web service.
  tags:
    - openEO - Secondary Services (OGC APIs)
  security:
    - Bearer: []
  responses:
    '200':
      description: Details of the created service
      content:
        application/json:
          schema:
            type: object
            required:
              - process
              - configuration
              - attributes
allOf:
  - $ref: '#/components/schemas/service'

4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
  $ref: '#/components/responses/server_error'

delete:
  summary: Delete a service
  operationId: delete-service
  description: Deletes all data related to this secondary web service. Computations are stopped, computed results are deleted and access to this is not possible any more. This service won't generate additional costs.
  tags: - openEO - Secondary Services (OGC APIs)
  security: - Bearer: [
  responses:
    '204':
      description: The service has been successfully deleted.
    4XX:
      $ref: '#/components/responses/client_error_auth'
    5XX:
      $ref: '#/components/responses/server_error'

'/services/{service_id}/logs':
  get:
    summary: Logs for a secondary service
    operationId: debug-service
    description: Lists log entries for the secondary service, usually for debugging purposes.
    tags: - openEO - Secondary Services (OGC APIs)
    security: - Bearer: [
    parameters:
      - $ref: '#/components/parameters/service_id'
      - $ref: '#/components/parameters/log_offset'
      - $ref: '#/components/parameters/log_level'
      - $ref: '#/components/parameters/pagination_limit'
    responses:
      '200':
        $ref: '#/components/responses/logs'
      4XX:
        $ref: '#/components/responses/client_error_auth'
      5XX:
        $ref: '#/components/responses/server_error'

/jobs:
  get:
summary: List all batch jobs
operationId: list-jobs
description: |
  Lists all batch jobs submitted by a user.

  It is **strongly RECOMMENDED** to keep the response size small by omitting all optional non-scalar values (i.e. arrays and objects) from objects in `jobs`. To get the full metadata for a job clients MUST request `GET /jobs/{job_id}`.

  **NOTE:** This endpoint may return jobs from openEO and OGC API - Processes. Distinguish them via the `jobID` (OGC API) and the `id` (openEO) property.
tags:
  - OGC API - Processes
  - openEO
security:
  - Bearer: []
parameters:
  - $ref: '#/components/parameters/pagination_limit'
responses:
  '200':
    description: Array of job descriptions
    content:
      application/json:
        schema:
          title: Batch Jobs
          type: object
          required:
            - jobs
            - links
          properties:
            jobs:
              type: array
              items:
                oneOf:
                  - title: openEO Batch Job
                    allOf:
                      - $ref: '#/components/schemas/batch_job'
                  - title: OGC API Job
                    allOf:
                      - $ref: '#/components/schemas/ogc_statusInfo'
            links:
              $ref: '#/components/schemas/links_pagination'
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'

post:
summary: Create a new batch job
operationId: create-job
description: |
  Creates a new batch processing task (job) from one or more (chained) processes at the back-end.

  Processing the data doesn't start yet. The job status gets initialized as `created` by default.
tags:
  - openEO
security:
  - Bearer: []
responses:
'201':
  description: The batch job has been created successfully.
headers:
  Location:
    required: true
    schema:
      description: `- Absolute URL to the newly created batch job.
        The URL points to the metadata endpoint
        `GET /jobs/{job_id}` with the `{job_id}` being the
        unique identifier (ID) of the created batch job.
      format: uri
type: string
element: example:
  example: 'https://geodatacube.example/api/v1/jobs/123'
GDC-Identifier:
  required: true
  schema:
    $ref: '#/components/schemas/job_id'
4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'
requestBody:
  required: true
  content:
    application/json:
      schema:
        title: Store Batch Job Request
        type: object
        required:
          - process
        properties:
          title:
            $ref: '#/components/schemas/eo_title'
          description:
            $ref: '#/components/schemas/eo_description'
          process:
            $ref: '#/components/schemas/process_graph_with_metadata'
          log_level:
            $ref: '#/components/schemas/min_log_level_default'
          additionalProperties:
            description: You can add additional back-end specific

properties here.
  description: 'Specifies the job details, e.g. the user-defined process
and billing details.'
'/jobs/{job_id}':
  parameters:
    - $ref: '#/components/parameters/job_id'
patch:
  summary: Modify a batch job
  operationId: update-job
  description: |-
    Modifies an existing job at the back-end, but maintains the identifier.
    Changes can be grouped in a single request.

    The job status does not change.

    Jobs can only be modified when the job is not queued and not running.
    Otherwise, requests to this endpoint MUST be rejected with an error.
tags:
  - openEO
security:
  - Bearer: []
responses:
  '204':
    description: Changes to the job applied successfully.
4XX:
    $ref: '#/components/responses/client_error_auth'
5XX:
    $ref: '#/components/responses/server_error'
requestBody:
  required: true
  content:
    application/json:
      schema:
        title: Update Batch Job Request
        type: object
        properties:
          title:
            $ref: '#/components/schemas/eo_title'
          description:
            $ref: '#/components/schemas/eo_description'
          process:
            $ref: '#/components/schemas/process_graph_with_metadata'
          log_level:
            $ref: '#/components/schemas/min_log_level_update'
        description: Specifies the job details to update.

get:
  summary: Full metadata for a batch job
  operationId: describe-job
  description: Lists information about a batch job.
  **NOTE:** This endpoint may return a job from openEO or OGC API - Processes. Distinguish them via the `jobID` (OGC API) and the `id` (openEO) property.
  tags:
    - openEO
    - OGC API - Processes
  security:
    - Bearer: []
  responses:
    '200':
      description: Full job information.
      content:
        application/json:
          schema:
            oneOf:
              - title: openEO Batch Job
                type: object
                required:
                - process
                allOf:
                  - $ref: '#/components/schemas/batch_job'
              - title: OGC API Job
                allOf:
                  - $ref: '#/components/schemas/ogc_statusInfo'
4XX:
    $ref: '#/components/responses/client_error_auth'
5XX:
    $ref: '#/components/responses/server_error'
delete:
  summary: Delete a batch job
operationId: delete-job
description: >-
  Deletes all data related to this job. Computations are stopped and
  computed results are deleted. This job won't generate additional costs
  for processing.
tags:
  - openEO
  - OGC API - Processes
security:
  - Bearer: []
responses:
  '200':
    description: The job has been successfully deleted (OGC API -
                Processes).
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ogc_statusInfo'
  '204':
    description: The job has been successfully deleted (openEO).
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'
'/jobs/{job_id}/logs':
get:
  summary: Logs for a batch job
  operationId: debug-job
  description: |-
    Lists log entries for the batch job, usually for debugging purposes.
    Back-ends can log any information that may be relevant for a user
    at any stage (status) of the batch job.
    Users can log information during data processing using respective
    processes such as `inspect`.
    If requested consecutively, it is RECOMMENDED that clients use the
    offset parameter to get only the entries they have not received yet.
    While pagination itself is OPTIONAL, the `offset` parameter is REQUIRED
    to be implemented by back-ends.
tags:
  - openEO
security:
  - Bearer: []
parameters:
  - $ref: '#/components/parameters/job_id'
  - $ref: '#/components/parameters/log_offset'
  - $ref: '#/components/parameters/log_level'
  - $ref: '#/components/parameters/pagination_limit'
responses:
  '200':
    $ref: '#/components/responses/logs'
  4XX:
    $ref: '#/components/responses/client_error_auth'
  5XX:
    $ref: '#/components/responses/server_error'
'/jobs/{job_id}/results':
get:
  summary: List batch job results
operationId: list-results

description: **NOTE:** This endpoint may return a job from openEO or OGC API - Processes. Distinguish them via the `assets` property which is always present for openEO, but not for OGC API - Processes.

## OGC API - Processes
Lists available results of a job. In case of a failure, lists exceptions instead.

For more information, see [Section 7.13](https://docs.ogc.org/is/18-062/18-062.html#sc_retrieve_job_results).

## openEO
Lists signed URLs pointing to the processed files, usually after the batch job has finished. Back-ends may also point to intermediate results after the job has stopped due to an error or if the `partial` parameter has been set.

The response includes additional metadata. It is a valid [STAC Item](https://github.com/radiantearth/stac-spec/tree/v1.0.0/item-spec) (if it has spatial and temporal references included) or a valid [STAC Collection](https://github.com/radiantearth/stac-spec/tree/v1.0.0/collection-spec). The assets to download are in both cases available in the property `assets` and have the same structure. All additional metadata is not strictly required to download the files, but are helpful for users to understand the data.

STAC Collections can either (1) add all assets as collection-level assets or (2) link to STAC Catalogs and STAC Items with signed URLs, which will provide a full STAC catalog structure a client has to go through. Option 2 is overall the better architectural choice and allows a fine-grained description of the processed data.

Clients are RECOMMENDED to store this response and all potential sub-catalogs and items with the assets so that the downloaded data is then a self-contained STAC catalog user could publish easily with all the data and metadata.

URL signing is a way to protect files from unauthorized access with a key in the URL instead of HTTP header based authorization. The URL signing key is similar to a password and its inclusion in the URL allows to download files using simple GET requests supported by a wide range of programs, e.g. web browsers or download managers. Back-ends are responsible to generate the URL signing keys and to manage their appropriate expiration. The back-end MAY indicate an expiration time by setting the `expires` property in the reponse. Requesting this endpoint SHOULD always return non-expired URLs. Signed URLs that were generated for a previous request and already expired SHOULD NOT be reused, but regenerated with new expiration time.
Signed URLs that expired MAY return an error.

It is **strongly recommended** to add a link with relation type `canonical` to the STAC Item or STAC Collection (see the `links` property for details).

If processing has not finished yet and the `partial` parameter is not set to `true`, requests to this endpoint MUST be rejected an error.

**openEO only**: If set to `true`, the results endpoint returns incomplete results while still running.

```json
parameters:
  - name: partial
description: >-
  **openEO only**: If set to `true`, the results endpoint returns incomplete results while still running.
in: query
allowEmptyValue: true
schema:
  type: boolean
default: false
responses:
  '200':
description: >-
  Provides the results.
content:
  application/json:
schema:
  oneOf:
  - $ref: '#/components/schemas/ogc_results'
  - $ref: '#/components/schemas/batch_job_result'
  - title: Batch Job Results Response as STAC Collection
type: object
required:
  - assets
allOf:
  - $ref: '#/components/schemas/collection'
example:
stac_version: 1.0.0
id: a3cca2b2aa1e3b5b
title: NDVI based on Sentinel 2
description: Deriving minimum NDVI measurements over pixel time series of Sentinel 2
license: Apache-2.0
providers:
  - name: Example Cloud Corp.
description: No further processing applied.
roles:
  - producer
  - licensor
  - host
url: https://cloud.example
extent:
temporal:
  interval:
    - 2019-08-24T14:15:22Z
    - 2019-08-24T14:15:22Z
spatial:
424:

description: >-
    The request can't be fulfilled as the batch job failed. This
    request will deliver the last error message that was produced by the batch
    job.
This HTTP code MUST be sent only when the job `status` is `error`.

```json
content:
  application/json:
    schema:
      $ref: '#/components/schemas/log_entry'

4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
  $ref: '#/components/responses/server_error'
```

**post:**

**summary:** Start processing a batch job

**operationId:** start-job

**description:**

Adds a batch job to the processing queue to compute the results.

The result will be stored in the format specified in the process. To specify the format use a process such as `save_result`.

The job status is set to `queued`, if processing doesn't start instantly. The same applies if the job status is `canceled`, `finished`, or `error`, which restarts the job and discards previous results if the back-end doesn't reject the request with an error.

Clients SHOULD warn users and ask for confirmation if results may get discarded.

* Once the processing starts the status is set to `running`.
* Once the data is available to download the status is set to `finished`.
* Whenever an error occurs during processing, the status MUST be set to `error`.

This endpoint has no effect if the job status is already `queued` or `running`. In particular, it doesn't restart a running job. To restart a queued or running job, processing MUST have been canceled before.

**tags:**

- openEO

**security:**

- Bearer: []

**responses:**

`202`:

**description:** The creation of the resource has been queued successfully.

```json
4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
  $ref: '#/components/responses/server_error'
```

**delete:**

**summary:** Cancel processing a batch job

**operationId:** stop-job

**description:**

 Cancels all related computations for this job at the back-end. It will stop generating additional costs for processing.

A subset of processed results may be available for downloading depending on the state of the job at the time it was canceled.

Results MUST NOT be deleted until the job processing is started again or the job is completely deleted through a request to
`DELETE /jobs/{job_id}`.

This endpoint only has an effect if the job status is `queued` or `running`.

The job status is set to `canceled` if the status was `running` beforehand and partial or preliminary results are available to be downloaded. Otherwise the status is set to `created`.

tags:
- openEO

security:
- Bearer: []

responses:
'204':
  description: Processing the job has been successfully canceled.

4XX:
  $ref: '#/components/responses/client_error_auth'

5XX:
  $ref: '#/components/responses/server_error'

/me:

get:
  summary: Information about the authenticated user
  operationId: describe-account
  description: >-
    Lists information about the authenticated user, e.g. the user id.

    The endpoint MAY return the disk quota available to the user. The endpoint MAY also return links related to user management and the user profile, e.g. where payments are handled or the user profile could be edited.

    This endpoint MAY be extended to fulfil the specification of the [OpenID Connect UserInfo Endpoint](http://openid.net/specs/openid-connect-core-1_0.html#UserInfo).

  tags:
  - Account Management

  security:
  - Bearer: []

  responses:
'200':
  description: Information about the logged in user.
  content:
    application/json:
      schema:
        title: User Data
        description: >-
          Holds user information.
        type: object
        required:
          - user_id
        properties:
          user_id:
            type: string
            description: >-
              A unique user identifier specific to the back-end, which could either be chosen by a user or is automatically generated by the back-end during the registration process at the back-end.

            It is meant to be used as an identifier in URIs (e.g. for sharing purposes), which is primarily used in machine-to-machine
communication. Preferably use the human-readable property `name` to display the user's name in user interfaces instead of the user identifier.

- **name:**
  
  type: string

  description: >-
  The user name, a human-friendly displayable name. Could be the user's real name or a nickname.

- **storage:**

  title: User Storage

  description: Information about the storage space available to the user.

  type: object

  nullable: true

  required:
  - free
  - quota

  properties:

  - **free:**

    type: integer

    description: >-
    Free storage space in bytes, which is still available to the user. Effectively, this is the disk quota minus the used space by the user, e.g. user-uploaded files and job results.

    example: 536870912

  - **quota:**

    type: integer

    description: >-
    Maximum storage space (disk quota) in bytes available to the user.

    example: 1073741824

- **links:**

  description: |-
  Links related to the user profile, e.g. where payments are handled or the user profile could be edited.

  It is RECOMMENDED to provide links with the following `rel` (relation) types:

  1. `edit-form`: Points to a page where the user can edit his user profile.

  2. `alternate`: Any other representation of these (and potentially additional) user information, e.g. the (public) user profile page.

  It is RECOMMENDED to add descriptive titles for a better user experience.

  3. `related`: Any other user-specific links to be shown in clients, RECOMMENDED to add descriptive titles for a better user experience.

  For additional relation types see also the lists of [common relation types](#section/API-Principles/Web-Linking).
$ref: '#/components/schemas/link'
example:
  - href: 'https://geodatacube.example/john_doe/payment/
    rel: payment
  - href: 'https://geodatacube.example/john_doe/edit/
    rel: edit-form
  - href: 'https://geodatacube.example/john_doe/
    rel: alternate
type: text/html
title: User profile
  - href: 'https://geodatacube.example/john_doe.vcf'
    rel: alternate
type: text/vcard
title: vCard of John Doe
  - href: 'https://geodatacube.example/john_doe/invoices'
    rel: related
type: text/html
title: Invoices

4XX:
  $ref: '#/components/responses/client_error_auth'
5XX:
  $ref: '#/components/responses/server_error'

components:
schemas:
  ogc_processSummary:
    allOf:
      - $ref: '#/components/schemas/descriptionType'
      - required:
        - id
        - version
    type: object
    properties:
      id:
        type: string
      version:
        type: string
    jobControlOptions:
      type: array
      items:
        $ref: '#/components/schemas/jobControlOptions'
    outputTransmission:
      type: array
      items:
        $ref: '#/components/schemas/transmissionMode'
    links:
      type: array
      items:
        $ref: '#/components/schemas/link'
  ogc_process:
    allOf:
      - $ref: '#/components/schemas/ogc_processSummary'
      - type: object
    properties:
      inputs:
        type: object
        additionalProperties:
          $ref: '#/components/schemas/inputDescription'
      outputs:
        type: object
        additionalProperties:
          $ref: '#/components/schemas/outputDescription'
  ogc_execute:
    type: object
properties:
  inputs:
    type: object
    additionalProperties:
      oneOf:
        - $ref: '#/components/schemas/inlineOrRefData'
        - type: array
          items:
            $ref: '#/components/schemas/inlineOrRefData'
  outputs:
    type: object
    additionalProperties:
      $ref: '#/components/schemas/ogc_output'
  response:
    type: string
    default: raw
    enum:
      - raw
      - document
  subscriber:
    $ref: '#/components/schemas/ogc_subscriber'
ogc_results:
  title: OGC API Results
  type: object
  additionalProperties:
    $ref: '#/components/schemas/inlineOrRefData'
ogc_statusInfo:
  required:
  - jobID
  - status
  - type
  type: object
  properties:
    processID:
      type: string
    jobID:
      type: string
    status:
      $ref: '#/components/schemas/ogc_statusCode'
    message:
      type: string
    created:
      type: string
      format: date-time
    started:
      type: string
      format: date-time
    finished:
      type: string
      format: date-time
    updated:
      type: string
      format: date-time
    progress:
      maximum: 100
      minimum: 0
      type: integer
    links:
      type: array
items:
  $ref: '#/components/schemas/link'

ogc_statusCode:
  type: string
  nullable: false
  enum:
    - accepted
    - running
    - successful
    - failed
    - dismissed

ogc_output:
  type: object
  properties:
    format:
      $ref: '#/components/schemas/ogc_format'
    transmissionMode:
      $ref: '#/components/schemas/transmissionMode'

ogc_format:
  type: object
  properties:
    mediaType:
      type: string
    encoding:
      type: string
    schema:
      oneOf:
        - type: string
          format: url
        - type: object

ogc_subscriber:
  required:
    - successUrl
  type: object
  properties:
    successUri:
      type: string
      format: uri
    inProgressUri:
      type: string
      format: uri
    failedUri:
      type: string
      format: uri
  description: Optional URIs for callbacks for this job.

  Support for this parameter is not required and the parameter may be removed from the API definition, if conformance class **'callback'** is not listed in the conformance declaration under `/conformance`

inlineOrRefData:
  oneOf:
    - $ref: '#/components/schemas/inputValueNoObject'
    - $ref: '#/components/schemas/qualifiedInputValue'
    - $ref: '#/components/schemas/link'

inputValue:
  oneOf:
    - $ref: '#/components/schemas/inputValueNoObject'
    - type: object

inputValueNoObject:
  oneOf:
    - type: string
    - type: number
- type: integer
- type: boolean
- type: array
  items:
    type: string
- $ref: '#/components/schemas/binaryInputValue'
- $ref: '#/components/schemas/bbox'

**binaryInputValue:**
- type: string
  format: byte

**qualifiedInputValue:**
- allOf:
  - $ref: '#/components/schemas/ogc_format'
  - required: [value]
    - type: object
      properties:
        value:
          $ref: '#/components/schemas/inputValue'

**inputDescription:**
- allOf:
  - $ref: '#/components/schemas/descriptionType'
  - required: [schema]
    - type: object
      properties:
        minOccurs:
          type: integer
          default: 1
        maxOccurs:
          oneOf:
            - type: integer
              default: 1
            - type: string
              enum:
                - unbounded
          schema:
            $ref: '#/components/schemas/json_schema'

**outputDescription:**
- allOf:
  - $ref: '#/components/schemas/descriptionType'
  - required: [schema]
    - type: object
      properties:
        schema:
          $ref: '#/components/schemas/json_schema'

**descriptionType:**
- type: object
  properties:
    title:
      type: string
    description:
      type: string
    keywords:
      type: array
      items:
        type: string
    metadata:
      type: array
      items:
        $ref: '#/components/schemas/metadata'

**additionalParameters:**
allOf:
  - $ref: '#/components/schemas/metadata'
  - type: object
    properties:
      parameters:
        type: array
        items:
          $ref: '#/components/schemas/additionalParameter'
  jobControlOptions:
    type: string
    enum:
      - sync-execute
      - async-execute
      - dismiss
  transmissionMode:
    type: string
    default: value
    enum:
      - value
      - reference
  metadata:
    type: object
    properties:
      title:
        type: string
      role:
        type: string
      href:
        type: string
  additionalParameter:
    required:
      - name
      - value
    type: object
    properties:
      name:
        type: string
      value:
        type: array
        items:
          oneOf:
            - type: string
            - type: number
            - type: integer
            - type: array
              items:
                type: object
      extent-uad:
        title: Extent with Uniform Additional Dimensions Schema
        description: |
          The extent module only addresses spatial and temporal extents. This module extends extent by specifying how intervals and crs properties can be used to specify additional geometries.
        allOf:
          - "$ref": "#/components/schemas/extent"
One or more intervals that describe the extent for this dimension of the dataset. The value `null` is supported and indicates an unbounded or half-bounded interval. The first interval describes the overall extent of the data for this dimension. All subsequent intervals describe more precise intervals, e.g., to identify clusters of data. Clients only interested in the overall extent will only need to access the first item (a pair of lower and upper bound values).

```json
    type: array
    minItems: 1
    items:
        description: |
          Lower and upper bound values of the interval. The values are in the coordinate reference system specified in `crs`, `trs` or `vrs`.

    type: array
    minItems: 2
    maxItems: 2
    items:
        oneOf:
            - type: string
              nullable: true
            - type: number

crs:
    type: string
    description: generic coordinate reference system suitable for any type of dimensions

trs:
    type: string
    description: temporal coordinate reference system (e.g. as defined by Features for 'temporal')

vrs:
    type: string
    description: vertical coordinate reference system (e.g. as defined in EDR for 'vertical')
```

```json
    crs:
        title: CRS
        oneOf:
            - description: Simplification of the object into a url if the other properties are not present
              type: string
            - type: object
              oneOf:
```

```json``
- required:
- uri
  properties:
  uri:
    description: Reference to one coordinate reference system (CRS)
    type: string
    format: uri
- required:
- wkt
  properties:
  wkt:
    description: A string defining the CRS using the JSON encoding
  for Well Known Text
  type: object
- required:
- referenceSystem
  properties:
  referenceSystem:
    description: A reference system data structure as defined in the
    MD_ReferenceSystem
    of the ISO 19115
    type: object
dataType:
  oneOf:
  - type: string
  - type: string
    enum:
    - map
    - vector
    - coverage
domainSet:
  type: object
  title: domainSet
  description: The domainSet describes the *direct positions* of the
  coverage, i.e., the locations for which values are available.
  oneOf:
  - required:
  - type
generalGrid:
  properties:
  type:
    enum:
    - DomainSet
generalGrid:
  title: General Grid
  description: A general n-D grid is defined through a sequence of
  axes, each of which can be of a particular axis type.
type: object
required:
- type
additionalProperties: false
properties:
  type:
    enum:
    - GeneralGridCoverage
id:
  type: string
srsName:
  type: string
format: uri
axisLabels:
  type: array
  items:
    type: string
axis:
  type: array
  items:
    type: object
    oneOf:
      - title: Index Axis
        description: An Index Axis is an axis with only integer positions allowed.
        required:
        - type
        - axisLabel
        - lowerBound
        - upperBound
        additionalProperties: false
        properties:
          type:
            enum:
              - IndexAxis
          id:
            type: string
          axisLabel:
            type: string
          lowerBound:
            type: number
          upperBound:
            type: number

      - title: Regular Axis
        description: A Regular Axis is an axis where all direct coordinates are at a common distance from its immediate neighbors.
        required:
        - type
        - axisLabel
        - lowerBound
        - upperBound
        - resolution
        - uomLabel
        additionalProperties: false
        properties:
          type:
            enum:
              - RegularAxis
          id:
            type: string
          axisLabel:
            type: string
          lowerBound:
            type: string
          upperBound:
            type: string
          resolution:
            type: number
          uomLabel:
            type: string

      - title: Irregular Axis
        description: An irregular axis enumerates all possible direct position coordinates.
required:
- type
- axisLabel
- uomLabel
- coordinate
additionalProperties: false
properties:
type:
  enum:
  - IrregularAxis
id:
  type: string
axisLabel:
  type: string
uomLabel:
  type: string
coordinate:
  type: array
  items:
    type: string
displacement:
title: Displacement
description: A Displacement is a warped axis nest where points on the grid all have their individual direct position coordinates.
The sequenceRule element describes linearization order.
type: object
oneOf:
- required:
  - type
  - axisLabels
  - uomLabels
  - coordinates
properties:
type:
  enum:
  - DisplacementAxisNest
id:
  type: string
axisLabel:
  type: string
srsName:
  type: string
format: uri
axisLabels:
  type: array
  items:
    type: string
uomLabels:
  type: array
  items:
    type: string
coordinates:
  type: array
  items:
    type: array
    items:
      type: string
- required:
  - type
  - axisLabels
  - uomLabels
- coordinatesRef
  properties:
  type: enum:
    enum:
      - DisplacementAxisNestRef
  id:
    type: string
  axisLabel:
    type: string
  srsName:
    type: string
  format: uri
  axisLabels:
    type: array
  items:
    type: string
  uomLabels:
    type: array
  items:
    type: string
  coordinatesRef:
    type: string
  format: uri

model:
  title: Sensor model
description: A Transformation By Sensor Model is a
transformation definition which is given by a SensorML 2.0 transformation
specification.
type: object
required:
  - type
  - sensorModelRef

properties:
  type: enum:
    enum:
      - TransformationBySensorModel
  id:
    type: string
  axisLabels:
    type: array
  items:
    type: string
  uomLabels:
    type: array
  items:
    type: string
  sensorModelRef:
    type: string
  format: uri
  sensorInstanceRef:
    type: string
  format: uri

gridLimits:
  title: Grid limits
description: This is the boundary of the array underlying the
given by its diagonal corner points in integer _60_3D. The
limits can be omitted in case all axes are of type index
then it repeats the grid information in a redundant way. The
all sequence of the axisLabels attribute, which lists the axis labels of axisExtent elements in proper sequence, is to enforce axis also in XML systems which do not preserve document order.

type: object
required:
- type
properties:
  type:
  enum:
  - GridLimits
indexAxis:
  title: Index Axis
description: An Index Axis is an axis with only integer positions allowed.
  type: object
required:
  - type
  - lowerBound
  - upperBound
additionalProperties: false
properties:
  type:
  enum:
  - IndexAxis
id:
  type: string
axisLabel:
  type: string
lowerBound:
  type: number
upperBound:
  type: number
srsName:
  type: string
  format: uri
axisLabels:
  type: array
  items:
  type: string
- required:
  - type
  - directMultiPoint
properties:
  type:
  enum:
  - DomainSet
directMultiPoint:
  oneOf:
  - required:
    - type
    - coordinates
  properties:
  type:
  enum:
  - DirectMultiPoint
coordinates:
  type: array
  items:
  type: array
  items:
type: string
- required:
  - type
  - coordinatesRef
properties:
type:
  enum:
  - DirectMultiPointRef
coordinatesRef:
type: string
format: uri
- required:
  - type
  - fileReference
properties:
type:
  enum:
  - DomainSetRef
id:
type: string
format: uri
fileReference:
type: string
format: uri
rangeType:
title: rangeType
description: The rangeType element describes the structure and semantics of a coverage's range values, including (optionally) restrictions on the interpolation allowed on such values.
type: object
oneOf:
- required:
  - type
  - field
properties:
type:
  enum:
  - DataRecord
field:
type: array
items:
title: DataRecord field
description: e.g. Quantity or Count
type: object
required:
- type
properties:
type:
  enum:
  - Quantity
  - Count
id:
type: string
format: uri
name:
type: string
definition:
type: string
format: uri
uom:
title: units of measure
description: units of measure
type: object
required:
- type
- code
properties:
type:
  enum:
  - UnitReference
id:
  type: string
  format: uri
code:
  type: string
constraint:
title: Constraint
description: Constraint
type: object
required:
- type
properties:
type:
  enum:
  - AllowedValues
id:
  type: string
  format: uri
interval:
  type: array
  items:
    type: string
interpolationRestriction:
title: interpolationRestriction
description: Interpolation restriction
type: object
required:
- type
properties:
type:
  enum:
  - InterpolationRestriction
id:
  type: string
  format: uri
allowedInterpolation:
  type: array
  items:
    type: string
    format: uri
- required:
  - type
  - fileReference
properties:
type:
  enum:
  - RangeTypeRef
id:
  type: string
  format: uri
fileReference:
  type: string
  format: uri
rangeSet:
**rangeSet**

The `rangeSet` lists a value for each of the coverage's direct positions. Values resemble the *payload* information of some particular direct positions. Values can be composite (with a single nesting level, i.e.: composites always consist of atomics), or atomic (emulated through single-component composites) whereby the sequence, structure, and meaning of every value is defined through the rangeType. Values can be represented in-line or by reference to an external file which may have any suitable encoding.'

```json
{  
  title: "rangeSet",  
  description: '{The rangeSet lists a value for each of the coverage''s direct positions. Values resemble the *payload* information of some particular direct positions. Values can be composite (with a single nesting level, i.e.: composites always consist of atomics), or atomic (emulated through single-component composites) whereby the sequence, structure, and meaning of every value is defined through the rangeType. Values can be represented in-line or by reference to an external file which may have any suitable encoding.'},
  type: object,
  oneOf:  
  - required:
  - type  
    dataBlock  
    properties:  
      type: object,  
      required:  
      - type  
      - values  
      properties:  
        type: array,
        items:  
          type: string
      - required:
      - type  
      - fileReference  
      properties:  
        type: object,  
        required:  
        - type  
        - fileReference  
      type: array,
      items:  
        type: string
  coverageSchema:  
  title: Coverage object
  description: 'Component of OGC Coverage Implementation Schema 1.1. Last updated: 2016-may-18. Copyright (c) 2016 Open Geospatial Consortium, Inc. All Rights Reserved. To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.'
}
```

domainSet
- rangeSet
- rangeType
properties:
  id:
    type: string
  type:
    enum:
      - CoverageByDomainAndRange
  envelope:
    title: envelope
description: The envelope around a coverage is defined by the lower and upper bound of each axis, respectively. The purpose of the axisLabels attribute, which lists the axis labels of all axisExtent elements in proper sequence, is to enforce axis sequence also in XML systems which do not preserve document order.
type: object
required:
  - type
  - srsName
  - axisLabels
  - axis
properties:
  type:
    enum:
      - EnvelopeByAxis
    id:
      type: string
    srsName:
      type: string
      format: uri
    axisLabels:
      type: array
    items:
      type: string
    axis:
      type: array
    items:
      type: object
      required:
        - type
        - lowerBound
        - upperBound
        - uomLabel
      additionalProperties: false
properties:
  type:
    enum:
      - AxisExtent
    id:
      type: string
    axisLabel:
      type: string
    lowerBound:
      oneOf:
        - type: number
        - type: string
        nullable: true
        - type: boolean
upperBound:
  oneOf:
  - type: number
  - type: string
    nullable: true
  - type: boolean

uomLabel:
  type: string

domainSet:
  "$ref": "/components/schemas/domainSet"
rangeSet:
  "$ref": "/components/schemas/rangeSet"
rangeType:
  "$ref": "/components/schemas/rangeType"

metadata:
  title: Metadata
  description: The metadata element is a container of any (not further specified) information which should be transported along with the coverage on hand, such as domain-specific metadata.
type: object
  - required:
    - type
    - partitionSet
    - rangeType
  properties:
    id:
      type: string
    type:
      enum:
      - CoverageByPartitioning
    envelope:
      "$ref": "/components/schemas/coverageSchema/oneOf/0/properties/envelope"

partitionSet:
  title: Partitioning Set
  description: A partition describes how a coverage (*sub-coverage*) is located within referencing coverage (*super-coverage*). The can be represented by referencing a coverage id or a URL pointing to a coverage. Such sub-coverages referenced may be grouped into the super-coverage document, or reside remote, or mixed. As an additional alternative, single range values can be indicated verbatim, together with their direct position. All values must share an identical structure and conform to the rangeType definition.
type: object
  required:
  - type
  properties:
    partition:
      type: array
      items:
        type: object
oneOf:
  - required:
    - type
    - coverageRef
  properties:
    id:
      type: string
      type: enum
      enum:
        - PartitionRef
    envelope:
      "$ref": "#/components/schemas/coverageSchema/oneOf/0"
properties/envelope
coverageRef:
  type: string
  format: uri
  - required:
    - type
    - coverage
  properties:
    id:
      type: string
      type: enum
      enum:
        - Partition
    envelope:
      "$ref": "#/components/schemas/coverageSchema/oneOf/0"
properties/envelope
coverage:
  type: object
positionValuePair:
  type: array
  items:
    type: object
    required:
      - type
      - coordinate
      - value
    properties:
      id:
        type: string
        type: enum
        enum:
          - PVP
    coordinate:
      type: array
      items:
        oneOf:
          - type: number
          - type: string
          - type: boolean
    value:
      type: array
      items:
        oneOf:
          - type: number
          - type: string
          - type: boolean
          nullable: true
rangeType:
  "$ref": "#/components/schemas/rangeType"
metadata:
"$ref": "/components/schemas/coverageSchema/oneOf/0/properties/metadata"

tileSet:
  title: Tile Set Metadata
  description: A resource describing a tileset based on the OGC TileSet Metadata
  type: object
  required:
  - dataType
  - crs
  - links
  properties:
    title:
      description: A title for this tileset
      type: string
    description:
      description: Brief narrative description of this tile set
      type: string
    dataType:
      allOf:
      - description: Type of data represented in the tileset
        \"$ref\": "/components/schemas/dataType"
    crs:
      allOf:
      - description: Coordinate Reference System (CRS)
        \"$ref\": "/components/schemas/crs"
      - description: Reference to a Tile Matrix Set on an official source for Tile Matrix Sets such as the OGC NA definition server (http://www.opengis.net/def/tms/).
        type: string
        format: uri
    links:
      description: 'Links to related resources. Possible link "rel" values are: 
      "http://www.opengis.net/def/rel/ogc/1.0/dataset" for a URL pointing to the dataset, "item" for a URL template to get a tile; 
      "alternate" (e.g. "http://www.opengis.net/def/rel/ogc/1.0/tiling-scheme") for a definition of the TileMatrixSet; "http://www.opengis.net/def/rel/ogc/1.0/geodata" for pointing to a single collection (if the tileset represents a single collection)
      "$ref": "/components/schemas/link"
    tileMatrixSetLimits:
      description: Limits for the TileRow and TileCol values for each TileMatrix in the tileMatrixSet. If missing, there are no limits other that imposed by the TileMatrixSet. If present the TileMatrices listed
and the rest not available at all

- type: array
- items:
  - "$ref": "/components/schemas/tileMatrixLimits"

- epoch:
  - description: Epoch of the Coordinate Reference System (CRS)
  - type: number

- layers:
  - minItems: 1
  - type: array
  - items:
    - type: object
    - required:
      - id
      - dataType
    - properties:
      - title:
        - description: Title of this tile matrix set, normally used for display to a human
        - type: string
      - description: Brief narrative description of this tile matrix set, normally available for display to a human
        - type: string
      - keywords:
        - description: Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this layer
        - type: string
      - id:
        - description: Unique identifier of the Layer. Implementation of 'identifier'
        - type: string
      - dataType:
        - allOf:
          - description: Type of data represented in the layer
          - "$ref": "/components/schemas/dataType"
        - description: 'The geometry dimension of the features shown in this layer (0: points, 1: curves, 2: surfaces, 3: solids), unspecified: mixed or unknown'
        - type: integer
        - minimum: 0
        - maximum: 3
      - featureType:
        - description: Feature type identifier. Only applicable to layers of datatype 'geometries'
        - type: string
      - pointOfContact:
        - description: Useful information to contact the authors or custodians for the layer (e.g. e-mail address, a physical address, phone numbers, etc)
        - type: string
      - publisher:
        - description: Organization or individual responsible for making the
layer available
  type: string
theme:
  description: Category where the layer can be grouped
  type: string
crs:
  allOf:
    - description: Coordinate Reference System (CRS)
    - "$ref": "#/components/schemas/crs"
epoch:
  description: Epoch of the Coordinate Reference System (CRS)
  type: number
minScaleDenominator:
  description: Minimum scale denominator for usage of the layer
  type: number
maxScaleDenominator:
  description: Maximum scale denominator for usage of the layer
  type: number
minCellSize:
  description: Minimum cell size for usage of the layer
  type: number
maxCellSize:
  description: Maximum cell size for usage of the layer
  type: number
maxTileMatrix:
  description: TileMatrix identifier associated with the
minScaleDenominator
  type: string
minTileMatrix:
  description: TileMatrix identifier associated with the
maxScaleDenominator
  type: string
boundingBox:
  allOf:
    - description: Minimum bounding rectangle surrounding the layer
    - "$ref": "#/components/schemas/2DBoundingBox"
created:
  allOf:
    - description: When the layer was first produced
    - "$ref": "#/components/schemas/timeStamp"
updated:
  allOf:
    - description: Last layer change/revision
    - "$ref": "#/components/schemas/timeStamp"
style:
  allOf:
    - description: Style used to generate the layer in the tileset
    - "$ref": "#/components/schemas/tileSet/properties/style/allOf/1"
geoDataClasses:
  description: URI identifying a class of data contained in this
layer
  (useful to determine compatibility with styles or processes)
  type: array
  items:
    type: string
propertiesSchema:
  allOf:
    - description: Properties represented by the features in this
layer.
  
  Can be the attributes of a feature dataset (datatype=
  geometries
  or the rangeType of a coverage (datatype=coverage)
Attributes of the features or rangetypes of a coverage. Defined by a subset of the JSON Schema for the properties of a feature:

```json
  type: object
  required:
    - type
    - properties
  properties:
    type:
      type: string
      enum:
        - object
      required:
        description: Implements 'multiplicity' by citing property 'name' defined as 'additionalProperties'
    type: array
    minItems: 1
    items:
      type: string
    properties:
      type: object
      default: {}
      additionalProperties:
        description: No property names are defined but any name they should be described by JSON Schema. So 'additionalProperties' implements 'name'.
    type: object
    properties:
      title:
        type: string
        description: Implements 'description'
      type:
        type: string
        enum:
          - array
          - boolean
          - integer
          - 'null'
          - number
          - object
          - string
        enum:
          description: Implements 'acceptedValues'
      type: array
      minItems: 1
      items: {}
      uniqueItems: true
      format:
        description: Complements implementation of 'type'
      type: string
      contentMediaType:
        description: Implements 'mediaType'
      type: string
      maximum:
        description: Implements 'range'
      type: number
```
exclusiveMaximum:
  description: Implements 'range'
  type: number
minimum:
  description: Implements 'range'
  type: number
exclusiveMinimum:
  description: Implements 'range'
  type: number
pattern:
  type: string
  format: regex
maxItems:
  description: Implements 'upperMultiplicity'
  type: integer
  minimum: 0
minItems:
  description: Implements 'lowerMultiplicity'
  type: integer
  default: 0
observedProperty:
  type: string
observedPropertyURI:
  type: string
  format: uri
uom:
  type: string
uomURI:
  type: string
  format: uri
links:
  description: 'Links related to this layer. Possible link
  "rel" values are: 'geodata' for a URL pointing to the collection of
  geospatial data.'
  type: array
  minItems: 1
  items:
    "$ref": "#/components/schemas/link"
boundingBox:
  allOf:
    - description: Minimum bounding rectangle surrounding the tile
      matrix set,
      in the supported CRS
    - "$ref": "#/components/schemas/2DBoundingBox"
centerPoint:
  allOf:
    - description: Location of a tile that nicely represents the
      tileset. Implementations may use this center value to set the default location or to
      present a representative tile in a user interface
    - type: object
      required:
      - coordinates
    properties:
      coordinates:
        type: array
        minItems: 2
        maxItems: 2
type: number
crs: allOf:
  - description: Coordinate Reference System (CRS) of the coordinates
    "$ref": "/components/schemas/crs"
tileMatrix: description: TileMatrix identifier associated with the scaleDenominator
type: string
scaleDenominator: description: Scale denominator of the tile matrix selected
type: number
cellSize: description: Cell size of the tile matrix selected
type: number
style: allOf:
  - description: Style involving all layers used to generate the tileset
type: object
  required:
  - id
  properties:
    id: description: An identifier for this style. Implementation of 'identifier'
type: string
title: description: A title for this style
type: string
description: Brief narrative description of this style
type: string
keywords: description: keywords about this style
type: array
  items:
    type: string
links: description: 'Links to style related resources. Possible link values are: ''style'' for a URL pointing to the style description, ''styleSpec'' for a URL pointing to the specification or standard used to define the style.'
type: array
  minItems: 1
  items:
    "$ref": "/components/schemas/link"
license: description: License applicable to the tiles
type: string
accessConstraints: description: Restrictions on the availability of the Tile Set that the user needs to be aware of before using or redistributing the Tile Set
type: string
  default: unclassified
  enum:
    - unclassified
    - restricted
- confidential
- secret
- topSecret

keywords:
  description: keywords about this tileset
  type: array
  items:
    type: string

version:
  description: Version of the Tile Set. Changes if the data behind the
  tiles
  has been changed
  type: string

created:
  allOf:
    - description: When the Tile Set was first produced
      - "$ref": "#/components/schemas/timeStamp"

updated:
  allOf:
    - description: Last Tile Set change/revision
      - "$ref": "#/components/schemas/timeStamp"

pointOfContact:
  description: Useful information to contact the authors or custodians
  for
  the Tile Set
  type: string

mediaTypes:
  description: Media types available for the tiles
  type: array
  items:
    type: string

tileSet-item:
  title: Tile Set Metadata item
  description: A minimal tileset element for use within a list of tilesets
  linking
to full description of those tilesets.
  type: object
  required:
    - dataType
    - links
    - crs
  properties:
    title:
      description: A title for this tileset
      type: string
    dataType:
      allOf:
        - description: Type of data represented in the tileset
          - "$ref": "#/components/schemas/dataType"
    crs:
      allOf:
        - description: Coordinate Reference System (CRS)
          - "$ref": "#/components/schemas/crs"

    tileMatrixSetURI:
      description: Reference to a Tile Matrix Set on official source for
      Tile
      Matrix Sets such as the OGC NA definition server (http://www.
      opengis.net/def/tms/).
      Required if the tile matrix set is registered on open official
      source.
      type: string
      format: uri
    links:
      type: array
      items:
description: Links to related resources. A 'self' link to the
  tileset as well as a 'http://www.opengis.net/def/rel/ogc/1.0/tiling-scheme'
  link to a definition of the TileMatrixSet are required.

tileMatrixLimits:
  title: TileMatrixLimits
  description: A resource describing useful to create an array that
describes the limits for a tile set TileMatrixSet based on the OGC TileSet
Metadata
  Standard
  type: object
  required:
  - tileMatrix
  - minTileRow
  - maxTileRow
  - minTileCol
  - maxTileCol
  properties:
    tileMatrix:
      type: string
    minTileRow:
      type: number
      format: integer
      minimum: 0
    maxTileRow:
      type: number
      format: integer
      minimum: 0
    minTileCol:
      type: number
      format: integer
      minimum: 0
    maxTileCol:
      type: number
      format: integer
      minimum: 0

2DPoint:
  description: A 2D Point in the CRS indicated elsewhere
  type: array
  minItems: 2
  maxItems: 2
  items:
    type: number

2DBoundingBox:
  description: Minimum bounding rectangle surrounding a 2D resource in the
  CRS indicated elsewhere
  type: object
  required:
  - lowerLeft
  - upperRight
  properties:
    lowerLeft:
      "$ref": "#/components/schemas/2DPoint"
    upperRight:
      "$ref": "#/components/schemas/2DPoint"
    crs:
      "$ref": "#/components/schemas/crs"
orderedAxes:
  type: array
  minItems: 2
  maxItems: 2
  items:
    type: string

tileMatrixSets:
  type: string
  enum:
    - WebMercatorQuad
    - WorldCRS84Quad
    - GNOSISGlobalGrid
    - WorldMercatorWGS84Quad

numberMatched:
  description: |
    The number of features of the feature type that match the selection
    parameters like `bbox`.
  type: integer
  minimum: 0
  example: 127

numberReturned:
  description: |
    The number of features in the feature collection.
    A server may omit this information in a response, if the information
    about the number of features is not known or difficult to compute.
    If the value is provided, the value must be identical to the number
    of items in the "features" array.
  type: integer
  minimum: 0
  example: 10

timeStamp:
  description: This property indicates the time and date when the response
    was generated.
  type: string
  format: date-time
  example: "2017-08-17T08:05:32Z"

conformsTo:
  description: |
    Lists all conformance classes specified in various standards that the
    implementation conforms to. Conformance classes are commonly used in
    all OGC APIs and the STAC API specification.
  type: array
  items:
    type: string
    format: uri
  example:
    - https://api.geodatascience.example/1.0.0-beta
    - https://api.stacspec.org/v1.0.0/core
    - https://api.stacspec.org/v1.0.0/collections
    - https://api.stacspec.org/v1.0.0/ogcapi-features
    - http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core
    - http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/json
    - http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas30
    - http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections
    - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core
    - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30
    - http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/geojson
    - http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-coverage
    - http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/cisjson
The GeoJSON type that applies to this metadata document, which MUST always be a "Feature" according to the STAC specification. This type does **not** describe the spatial data type of the assets.

Results without a known location can set this value to `null`.

```
example:
type: Polygon
coordinates:
- - - -180
  - -90
  - -180
  - -90
  - 90
  - -180
  - 90
  - -90
```

```
properties:
datetime:
title: Date and Time
description: The searchable date/time of the data, in UTC. Formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.

If this field is set to `null` (usually for larger time ranges), it is STRONGLY RECOMMENDED to specify both `start_datetime` and `end_datetime` for STAC compliance.
```
For time series: The first or start date and time for the data, in UTC. Formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.

`end_datetime`:
- `type`: string
- `format`: date-time
- `description`: For time series: The last or end date and time for the data, in UTC. Formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.

`title`:
- `$ref`: '#/components/schemas/eo_title'

`description`:
- `$ref`: '#/components/schemas/eo_description'

`license`:
- `$ref`: '#/components/schemas/stac_license'

`providers`:
- `$ref`: '#/components/schemas/stac_providers'

`created`:
- `$ref`: '#/components/schemas/created'

`updated`:
- `$ref`: '#/components/schemas/updated'

`expires`:
- `type`: string
- `format`: date-time
- `description`: Time until which the assets are accessible, in UTC. Formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.

`example`: '2020-11-01T00:00:00Z'

`stac_item`:
- `title`: A STAC Item
- `description`: The STAC specification should be the main guidance for implementing this.
- `type`: object
- `required`:
  - `stac_version`
  - `id`
  - `type`
  - `geometry`
  - `properties`
  - `assets`
  - `links`

`properties`:
- `stac_version`:
  - `$ref`: '#/components/schemas/stac_version'

- `stac_extensions`:
  - `$ref`: '#/components/schemas/stac_extensions'

- `id`:
  - `type`: string

- `type`:
  - `$ref`: '#/components/schemas/stac_item_type'

- `bbox`:
  - `$ref`: '#/components/schemas/bbox'

- `geometry`:
  - `$ref`: '#/components/schemas/stac_item_geometry'

- `properties`:
  - `$ref`: '#/components/schemas/stac_item_properties'

- `assets`:
  - `$ref`: '#/components/schemas/stac_assets'

- `links`:
  - `$ref`: '#/components/schemas/links'
batch_job_result:
  title: openEO - Batch Job Results Response as STAC Item
  description:
    The STAC specification should be the main guidance for implementing this.
    Specifying the `bbox` is strongly RECOMMENDED, but can be omitted if the result is unlocated and the `geometry` is set to `null`.
  type: object
  required:
    - stac_version
    - id
    - type
    - geometry
    - properties
    - assets
    - links
  properties:
    stac_version:
      $ref: '#/components/schemas/stac_version'
    stac_extensions:
      $ref: '#/components/schemas/stac_extensions'
    id:
      $ref: '#/components/schemas/job_id'
    type:
      $ref: '#/components/schemas/stac_item_type'
    geometry:
      $ref: '#/components/schemas/stac_item_geometry'
    properties:
      $ref: '#/components/schemas/stac_item_properties'
    assets:
      $ref: '#/components/schemas/stac_assets'
    links:
      type: array
      description: |
        Links related to this batch job result, e.g. a link to an invoice, additional log files or external documentation.
        The links MUST NOT contain links to the processed and downloadable data. Instead specify these in the `assets` property. Clients MUST NOT download the data referenced in the links by default.
        It is **strongly recommended** to add a link with relation type `canonical`, which points to this STAC document using a signed URL. This way the STAC metadata can be used by other clients without additional authentication steps.
        For relation types see the lists of [common relation types](#section/API-Principles/Web-Linking).
      items:
        $ref: '#/components/schemas/link'
  example:
    - rel: canonical
      type: application/geo+json
      href: https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/item.json

file_format:
  x-additionalPropertiesName: File Format Name
  title: File Format
type: object
description: Describes a specific file format.
required:
  - gis_data_types
  - parameters
properties:
title:
  $ref: '#/components/schemas/object_title'
description:
  $ref: '#/components/schemas/description'
gis_data_types:
  type: array
description: Specifies the supported GIS spatial data types for this format.
  minItems: 1
  items:
    type: string
    enum:
      - raster
      - vector
      - table
      - pointcloud
      - other
deprecated:
  $ref: '#/components/schemas/deprecated'
experimental:
  $ref: '#/components/schemas/experimental'
parameters:
  title: File Format Parameters
description: Specifies the supported parameters for this file format.
  type: object
  additionalProperties:
    $ref: '#/components/schemas/resource_parameter'
links:
  type: array
  description: Links related to this file format, e.g. external documentation.
    For relation types see the lists of [common relation types](#section/API-Principles/Web-Linking).
  items:
    $ref: '#/components/schemas/link'
links_pagination:
  description:
    Links related to this list of resources, for example links for pagination or alternative formats such as a human-readable HTML version. The links array MUST NOT be paginated.
    If pagination is implemented, the following `rel` (relation) types apply:
    1. `next` (REQUIRED): A link to the next page, except on the last page.
    2. `prev` (OPTIONAL): A link to the previous page, except on the first page.
    3. `first` (OPTIONAL): A link to the first page, except on the first page.
    4. `last` (OPTIONAL): A link to the last page, except on the last page.
   For additional relation types see also the lists of
[common relation types](#section/API-Principles/Web-Linking).

**type:** array

**items:**

$ref: '#/components/schemas/link'

**links:**

**description:**
- Links related to this list of resources, for example links for pagination or alternative formats such as a human-readable HTML version.

The links array MUST NOT be paginated.

For relation types see also the lists of [common relation types](#section/API-Principles/Web-Linking).

**link:**

**title:** Link

**description:**


**type:** object

**required:**
- href
- rel

**properties:**

- **rel:**
  - **type:** string
    - **description:**
      - Relationship between the current document and the linked document. SHOULD be a [registered link relation type](https://www.iana.org/assignments/link-relations/link-relations.xml) whenever feasible.
    - **example:** related
  - **href:**
    - **type:** string
      - **description:** The value MUST be a valid URL.

**format:** uri

**example:** 'https://geodatacube.example'

- **type:** string
  - **description:**
    - The value MUST be a string that hints at the format used to represent data at the provided URI, preferably a media (MIME) type.
    - **example:** text/html

- **title:**
  - **type:** string
    - **description:** Used as a human-readable label for a link.
    - **example:** Example title

**asset:**

- **title:** STAC Asset
- **type:** object

**required:**
- href

**properties:**

- **href:**
  - **title:** Asset location
  - **description:**
    - URL to the downloadable asset.

The URLs SHOULD be available without authentication so that external clients can download them easily.
If the data is confidential, signed URLs SHOULD be used to protect against unauthorized access from third parties.

type: string
title:

description: The displayed title for clients and users.
type: string
description:
type: string
format: commonmark
description: |
  Multi-line description to explain the asset.

[CommonMark 0.29](http://commonmark.org/) syntax MAY be used for rich
text representation.

type:
title: Media Type
description: Media type of the asset.
type: string
example: image/tiff; application=geotiff
roles:
type: array
items:
type: string
description: |
Purposes of the asset. Can be any value, but commonly used values are:

* `thumbnail`: A visualization of the data, usually a lower-resolution true color image in JPEG or PNG format.
* `reproducibility`: Information how the data was produced and/or can be reproduced, e.g. the process graph used to compute the data in JSON format.
* `data`: The computed data in the format specified by the user in the process graph (applicable in `GET /jobs/{job_id}/results` only).
* `metadata`: Additional metadata available for the computed data.

example:
- data
stac_extent:
  allOf:
  - $ref: '#/components/schemas/extent'
  - required:
    - spatial
    - temporal
extent:
type: object
title: Collection Extent
description: |
The extent of the data in the collection. Additional members MAY be added to represent other extents, for example, thermal or pressure ranges.

The first item in the array always describes the overall extent of the data. All subsequent items describe more precise extents, e.g. to identify clusters of data. Clients only interested in the overall extent will only need to access the first item in each array.

properties:
spatial:
title: Collection Spatial Extent
description: >-
The spatial extents of the data in the collection.
type: object
properties:
  bbox:
    description: |
      One or more bounding boxes that describe the spatial extent
      of the dataset.

      The first bounding box describes the overall spatial extent
      of the data. All subsequent bounding boxes describe more
      precise bounding boxes, e.g. to identify clusters of data.
      Clients only interested in the overall spatial extent will
      only need to access the first item in each array.
    type: array
    minItems: 1
    items: $ref: '#/components/schemas/bbox'
  crs:
    description: |
      Coordinate reference system of the coordinates in the spatial
      extent (property `bbox`). The default reference system is WGS 84
      longitude/latitude.
      In the Core the only other supported coordinate reference
      system is WGS 84 longitude/latitude/ellipsoidal height for coordinates
      with height.
      Extensions may support additional coordinate reference systems
      and add additional enum values.
    type: string
    enum:
      - http://www.opengis.net/def/crs/OGC/1.3/CRS84
      - http://www.opengis.net/def/crs/OGC/0/CRS84h
    default: http://www.opengis.net/def/crs/OGC/1.3/CRS84
  grid:
    description: |
      Provides information about the limited availability of data
      within the collection organized
      as a grid (regular or irregular) along each spatial dimension.
    type: array
    minItems: 2
    maxItems: 3
    items:
      type: object
      properties:
        coordinates:
          description: |
            List of coordinates along the dimension for which data
            organized as an irregular grid in the collection is available
            (e.g., 2, 10, 80, 100).
          type: array
          minItems: 1
          items:
            oneOf:
              - type: string
                nullable: true
              - type: number
            example:
              - 2
              - 10
              - 80
              - 100
        cellsCount:
description: |-
   Number of samples available along the dimension for data
   organized as a regular grid.
   For values representing the whole area of contiguous
   cells spanning _resolution_ units along the dimension, this will be (_
   upperBound_ - _lowerBound_) / _resolution_.
   For values representing infinitely small point cells
   spaced by _resolution_ units along the dimension, this will be (_upperBound_ -
   _lowerBound_) / _resolution_ + 1.
   type: integer
   example: 50
resolution:
   description: Resolution of regularly gridded data along
   the dimension in the collection
   oneOf:
   - type: string
     nullable: true
   - type: number
     example: 0.0006866455078
temporal:
  title: Collection TemporalExtent
  description: >-
    The temporal extents of the data in the
    collection.
  type: object
  properties:
    interval:
      description: |-
        One or more time intervals that describe the temporal extent
        of the dataset.
        The first time interval describes the overall temporal extent
        of the data. All subsequent time intervals describe more
        precise time intervals, e.g. to identify clusters of data.
        Clients only interested in the overall extent will only need
        to access the first item in each array.
      type: array
      minItems: 1
      items:
        description: |-
          Begin and end times of the time interval. The coordinate
          reference system is the Gregorian calendar.
          The value `null` is supported and indicates an open time
          interval.
        type: array
        minItems: 2
        maxItems: 2
        items:
          type: string
          format: date-time
          nullable: true
          example:
            - null
        trs:
          description: |-
            Coordinate reference system of the coordinates in the temporal
            extent (property `interval`). The default reference system is the
            Gregorian calendar.
            In the Core this is the only supported temporal coordinate
            reference system.
Extensions may support additional temporal coordinate reference systems and add additional enum values.

```json
    type: string
genre:
    
    - http://www.opengis.net/def/uom/ISO-8601/0/Gregorian
default: http://www.opengis.net/def/uom/ISO-8601/0/Gregorian
```

additionalProperties:

    description: The domain intervals for any additional dimensions of the extent (envelope) beyond those described in temporal and spatial.

```json
type: object
oneOf:
      - required:
        - interval
        - crs
      - required:
        - interval
        - trs
      - required:
        - interval
        - vrs
```

properties:

```json
interval:
    description: One or more intervals that describe the extent for this dimension of the dataset. The value 'null' is supported and indicates an unbounded or half-bounded interval. The first interval describes the overall extent of the data for this dimension. All subsequent intervals describe more precise intervals, e.g., to identify clusters of data. Clients only interested in the overall extent will only need to access the first item (a pair of lower and upper bound values).

type: array
minItems: 1
items:
    description: Lower and upper bound values of the interval. The values are in the coordinate reference system specified in `crs`, `trs` or `vrs`.

type: array
minItems: 2
maxItems: 2
items:
    oneOf:
        - type: string
          nullable: true
        - type: number
```

crs:
    type: string
description: generic coordinate reference system suitable for any type of dimensions

trs:
    type: string
description: temporal coordinate reference system (e.g. as defined by Features for 'temporal')

vrs:
    type: string
description: vertical coordinate reference system (e.g. as defined in EDR for 'vertical')

grid:
type: object
description: Provides information about the limited availability of data within the collection organized as a grid (regular or irregular) along the dimension.

properties:
  coordinates:
    description: List of coordinates along the temporal dimension for which data organized as an irregular grid in the collection is available (e.g., 2, 10, 80, 100).
    type: array
    minItems: 1
    items:
      oneOf:
        - type: string
          nullable: true
        - type: number
          example:
            - 2
            - 10
            - 80
            - 100

  cellsCount:
    description: Number of samples available along the dimension for data organized as a regular grid.
    type: integer
    example: 50

  resolution:
    description: Resolution of regularly gridded data along the dimension in the collection
    oneOf:
      - type: string
        nullable: true
      - type: number
        example:
          - PT1H
          - 0.0006866455078

collection:
  title: Coverages Collection
  type: object
  required:
    - id
    - extent
    - links
  properties:
    id:
      $ref: '#/components/schemas/collection_id'
    title:
      type: string
description: A short descriptive one-line title for the collection.
  description:
    type: string
    format: commonmark
description: Detailed multi-line description to explain the collection.
**CommonMark 0.29** syntax MAY be used for rich text representation.

**extent:**

`$ref: '#/components/schemas/extent'`

**links:**

- **description:** Links related to this collection. Could reference to licensing information, other meta data formats with additional information or a preview image.

It is RECOMMENDED to provide links with the following `rel` (relation) types:

1. `root` and `parent`: URL to the data discovery endpoint at `/collections`.
2. `license`: A link to the license(s) SHOULD be specified if the field is set to `proprietary` or `various`.
3. `example`: Links to examples of processes that use this collection.
4. `latest-version`: If a collection has been marked as deprecated, a link SHOULD point to the latest version of the collection. The relation types `predecessor-version` (link to older version) and `successor-version` (link to newer version) can also be used to show the relation between versions.
5. `alternate`: An alternative representation of the collection. For example, this could be the collection available through another catalog service such as OGC CSW, a human-readable HTML version or a metadata document following another standard such as ISO 19115 or DCAT.

For JSON Schema documents, the `type` field must be set to `application/schema+json`.

For additional relation types see also the lists of [common relation types](#section/API-Principles/Web-Linking) and the STAC specification for Collections.

**type:** array

**items:**

`$ref: '#/components/schemas/link'`

**itemType:**

- **description:** indicator about the type of the items in the collection if the collection has an accessible /collections/{collectionId}/items endpoint

**crs:**

- **description:** the list of coordinate reference systems supported by the API; the first item is the default coordinate reference system

**type:** array

**items:**

- **type:** string

**default:**

- `http://www.opengis.net/def/crs/OGC/1.3/CRS84`

**example:**

- `http://www.opengis.net/def/crs/OGC/1.3/CRS84`
dataType:
  allOf:
    - description: Type of data represented in the collection
      $ref: '#/components/schemas/dataType'
geometryDimension:
  description: The geometry dimension of the features shown in this layer (0: points, 1: curves, 2: surfaces, 3: solids), unspecified: mixed or unknown
    type: integer
    minimum: 0
    maximum: 3
minScaleDenominator:
  description: Minimum scale denominator for usage of the collection
  type: number
maxScaleDenominator:
  description: Maximum scale denominator for usage of the collection
  type: number
minCellSize:
  description: Minimum cell size for usage of the collection
  type: number
maxCellSize:
  description: Maximum cell size for usage of the collection
  type: number
stac_collection:
  title: STAC / openEO Collection
  type: object
  required:
    - stac_version
    - type
    - description
    - license
    - links
  properties:
    stac_version:
      $ref: '#/components/schemas/stac_version'
    stac_extensions:
      $ref: '#/components/schemas/stac_extensions'
      type: string
      enum:
        - Collection
    keywords:
      type: array
      description: List of keywords describing the collection.
      items:
        type: string
    license:
      $ref: '#/components/schemas/stac_license'
    providers:
      $ref: '#/components/schemas/stac_providers'
    extent:
      $ref: '#/components/schemas/stac_extent'
      'cube:dimensions':
        title: STAC Collection Cube Dimensions
        description: The named default dimensions of the data cube. Names must be unique per collection.
        - The keys of the object are the dimension names. For interoperability, it is RECOMMENDED to use the following dimension names if there is only a single dimension with the specified criteria:
This property **REQUIRES** to add a version of the data cube extension to the list of `stac_extensions`, e.g. `https://stac-extensions.github.io/datacube/v2.2.0/schema.json`.

```json
type: object
  additionalProperties:
    x-additionalPropertiesName: Dimension Name
  allOf:
    - $ref: '#/components/schemas/dimension'

summaries:
  title: STAC Summaries (Collection Properties)
  description: |
    Collection properties from STAC extensions (e.g. EO, SAR, Satellite or Scientific) or even custom extensions.

    Summaries are either a unique set of all available values, statistics *or* a JSON Schema. Statistics only specify the range (minimum and maximum values) by default, but can optionally be accompanied by additional statistical values. The range can specify the potential range of values, but it is recommended to be as precise as possible. The set of values MUST contain at least one element and it is strongly RECOMMENDED to list all values. It is recommended to list as many properties as reasonable so that consumers get a full overview of the Collection. Properties that are covered by the Collection specification (e.g. `providers` and `license`) SHOULD NOT be repeated in the summaries.

    Potential fields for the summaries can be found here:
    * **[STAC Common Metadata](https://github.com/radiantearth/stac-spec/blob/v1.0.0/item-spec/common-metadata.md)**:
      A list of commonly used fields throughout all domains
    * **[Content Extensions](https://github.com/radiantearth/stac-spec/blob/v1.0.0/extensions/README.md#list-of-content-extensions)**:
      Domain-specific fields for domains such as EO, SAR and point clouds.
    * **Custom Properties**:
      It is generally allowed to add custom fields.
```

```json
type: object
  additionalProperties:
    oneOf:
      - type: array
        title: Set of values
        items:
          description: A value of any type.
      - $ref: '#/components/schemas/collection_summary_stats'
      - $ref: '#/components/schemas/json_schema'

assets:
  description: |
    Dictionary of asset objects for data that can be downloaded, each with a unique key.
    The keys MAY be used by clients as file names.
allOf:
  - $ref: '#/components/schemas/stac_assets'

stac_version:
  type: string
  description: The version of the STAC specification, which MAY not be equal to the STAC API version.
  supports versions 1.x.x.
  pattern: '^1\.\d+\.\d+$'
  example: 1.0.0

stac_extensions:
  type: array
  description: A list of implemented STAC extensions. The list contains URLs to the JSON Schema files it can be validated against. For STAC < 1.0.0-rc.1 shortcuts such as `sar` can be used instead of the schema URL.
  uniqueItems: true
  items:
    anyOf:
      - title: Reference to a JSON Schema
        type: string
        format: uri
        example: 'https://geodatacube.example/stac/custom-extension/v1.0.0/schema.json'
      - title: Reference to a core extension (STAC < 1.0.0-rc.1 only)
        type: string
        example: datacube

stac_license:
  type: string
  description: License(s) of the data as a SPDX License identifier. Alternatively, use `proprietary` if the license is not on the SPDX license list or `various` if multiple licenses apply. In these two cases links to the license texts SHOULD be added, see the `license` link relation type.
  example: Apache-2.0

stac_providers:
  type: array
  description: A list of providers, which MAY include all organizations capturing or processing the data or the hosting provider. Providers SHOULD be listed in chronological order with the most recent provider being the last element of the list.
  items:
    type: object
    title: Provider
    required:
      - name
    properties:
      name:
        description: The name of the organization or the individual.
        type: string
        example: Example Cloud Corp.
        description:
description: >-
  Multi-line description to add further provider information such
  as
  processing details for processors and producers, hosting details
  for hosts or basic contact information.

  CommonMark 0.29 syntax MAY be used for rich text representation.
type: string
eexample: No further processing applied.
roles:
  description: |
  Roles of the provider.
  The provider's role(s) can be one or more of the following elements:
  * `licensor`: The organization that is licensing the dataset
    under the license specified in the collection's license field.
  * `producer`: The producer of the data is the provider that
    initially captured and processed the source data, e.g. ESA for
    Sentinel-2 data.
  * `processor`: A processor is any provider who processed data to
    a derived product.
  * `host`: The host is the actual provider offering the data on
    their storage. There SHOULD be no more than one host, specified as last
    element of the list.
type: array
items:
  type: string
enum:
  - producer
  - licensor
  - processor
  - host
eexample:
  - producer
  - licensor
  - host
url:
  description: >-
    Homepage on which the provider describes the dataset and
    publishes contact information.
type: string
format: uri
eexample: https://cloud.example
stac_assets:
  type: object
title: Assets
description: |
  Dictionary of asset objects for data that can be downloaded, each with
  a unique key. The keys MAY be used by clients as file names.
additionalProperties:
  $ref: '#/components/schemas/asset'
eexample:
  preview.png:
    href: 'https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/preview.png'
type: image/png
title: Thumbnail
roles:
- thumbnail
process.json:
  href: 'https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/process.json'
type: application/json
title: Original Process
roles:
- process
- reproduction
1.tif:
  href: 'https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/1.tif'
type: image/tiff; application=geotiff
title: Band 1
roles:
- data
2.tif:
  href: 'https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/2.tif'
type: image/tiff; application=geotiff
title: Band 2
roles:
- data
inspire.xml:
  href: 'https://geodatacube.example/api/v1/download/583fba8b2ce583fba8b2ce/inspire.xml'
type: application/xml
title: INSPIRE metadata
description: INSPIRE compliant XML metadata
roles:
- metadata
collection_summary_stats:
type: object
title: Statistics / Range
description: >-
  By default, only ranges with a minimum and a maximum value can be specified. Ranges can be
  specified for ordinal values only, which means they need to have a rank order. Therefore,
  ranges can only be specified for numbers and some special types of strings. Examples: grades
  (A to F), dates or times. Implementors are free to add other derived statistical values to the
  object, for example `mean` or `stddev`.
required:
- minimum
- maximum
properties:
  minimum:
    description: The minimum value (inclusive).
    anyOf:
      - type: string
      - type: number
  maximum:
    description: The maximum value (inclusive).
    anyOf:
      - type: string
      - type: number
bbox:
  description: |-
    Each bounding box is provided as four or six numbers,
depending on whether the coordinate reference system 
includes a vertical axis (height or depth):

* West (lower left corner, coordinate axis 1)
* South (lower left corner, coordinate axis 2)
* Base (optional, minimum value, coordinate axis 3)
* East (upper right corner, coordinate axis 1)
* North (upper right corner, coordinate axis 2)
* Height (optional, maximum value, coordinate axis 3)

The coordinate reference system of the values is WGS 84 
longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84).

For WGS 84 longitude/latitude the values are in most cases 
the sequence of minimum longitude, minimum latitude, maximum 
longitude and maximum latitude.

However, in cases where the box spans the antimeridian the 
first value (west-most box edge) is larger than the third value 
(east-most box edge).

If the vertical axis is included, the third and the sixth 
number are the bottom and the top of the 3-dimensional bounding box.

```
type: array
oneOf:
  - title: 4 elements
    minItems: 4
    maxItems: 4
  - title: 6 elements
    minItems: 6
    maxItems: 6
items:
  type: number
example:
  - 180
  - 90
  - 180
  - 90

collection_id:
  type: string
description: A unique identifier for the collection, which MUST match the specified 
pattern.
pattern: '^[\w\-\.~\/-]+$'
example: Sentinel-2A
dimension:
  title: Dimension
description: A dimension, each object represents a distinct dimension 
with the key being the dimension name.
type: object
required:
  - type
properties:
  type:
    description: Type of the dimension.
type: string
enum:
  - spatial
  - temporal
  - bands
  - geometry
  - other
description: 
```
$ref: '#/components/schemas/description'
discriminator:
  propertyName: type
mapping:
  spatial: '#/components/schemas/dimension_spatial'
  temporal: '#/components/schemas/dimension_temporal'
  bands: '#/components/schemas/dimension_bands'
  geometry: '#/components/schemas/dimension_geometry'
  other: '#/components/schemas/dimension_other'
dimension_other:
  allOf:
    - $ref: '#/components/schemas/dimension'
    - title: Additional Dimension
type: object
    oneOf:
      - title: Additional Dimension with Extent
        required:
        - extent
      - title: Additional Dimension with Values
        required:
        - values
    properties:
      extent:
        $ref: '#/components/schemas/collection_dimension_extent_open'
      values:
        $ref: '#/components/schemas/collection_dimension_values'
      step:
        $ref: '#/components/schemas/collection_dimension_step'
      unit:
        $ref: '#/components/schemas/collection_dimension_unit'
      reference_system:
        description: The reference system for the dimension.
type: string
dimension_geometry:
  allOf:
    - $ref: '#/components/schemas/dimension'
    - title: Geometry Dimension
type: object
    required:
    - bbox
    properties:
      axes:
        description: Axes of the vector dimension as an ordered set of `x`, `y` and `z`. Defaults to `x` and `y`.
        default:
        - `x`
        - `y`
type: array
        uniqueItems: true
        items:
          $ref: '#/components/schemas/dimension_axis_xyz'
        bbox:
          $ref: '#/components/schemas/bbox'
        values:
          description: Optionally, a representation of the vectors. This can be a list of WKT string or other free-form identifiers.
type: array
        items:
          type: string
        geometry_types:
          description: A set of all geometry types included in this dimension. If not present, mixed geometry types must be assumed.
type: array
uniqueItems: true
items:
  $ref: '#/components/schemas/geometry_type'
reference_system:
  $ref: '#/components/schemas/collection_dimension_srs'
dimension_bands:
  allOf:
    - $ref: '#/components/schemas/dimension'
    - title: Band Dimension
description: |
      A dimension for the bands.
The band dimension only specifies the band names as dimension labels. Further information to the bands are available in either `sar:bands` or `eo:bands` in the `summaries` property.
type: object
required:
  - values
properties:
  values:
    $ref: '#/components/schemas/collection_dimension_values'
dimension_spatial:
  allOf:
    - $ref: '#/components/schemas/dimension'
    - title: Spatial Dimension
description: A spatial (raster) dimension in one of the horizontal (x or y) or vertical (z) directions.
type: object
required:
  - axis
properties:
  axis:
    $ref: '#/components/schemas/dimension_axis_xyz'
extent:
  description: >-
    Extent (lower and upper bounds) of the dimension as two-dimensional array. Open intervals with `null` are not allowed.
type: array
minItems: 2
maxItems: 2
items:
  type: number
values:
  description: 'A set of all potential values.'
type: array
minItems: 1
items:
  type: number
step:
  $ref: '#/components/schemas/collection_dimension_step'
reference_system:
  $ref: '#/components/schemas/collection_dimension_srs'
discriminator:
  propertyName: axis
mapping:
  x: '#/components/schemas/dimension_spatial_horizontal'
  y: '#/components/schemas/dimension_spatial_horizontal'
  z: '#/components/schemas/dimension_spatial_vertical'
dimension_axis_xyz:
  title: Axis
description: Axis of a geometry or dimension (`x`, `y` or `z`)
type: string
enum:
  - 'x'
  - 'y'
  - 'z'
dimension_spatial_horizontal:
  allOf:
    - $ref: '#/components/schemas/dimension_spatial'
    - title: Horizontal Spatial Dimension
      required:
        - extent
dimension_spatial_vertical:
  allOf:
    - $ref: '#/components/schemas/dimension_spatial'
    - title: Vertical Spatial Dimension
      anyOf:
        - title: Vertical Spatial Dimension with Extent
          required:
            - extent
        - title: Vertical Spatial Dimension with Values
          required:
            - values
dimension_temporal:
  allOf:
    - $ref: '#/components/schemas/dimension'
    - title: Temporal Dimension
      description:>
        A temporal dimension based on the ISO 8601 standard. The temporal reference system for the data is expected to be ISO 8601 compliant (Gregorian calendar / UTC). Data not compliant with ISO 8601 can be represented as an *Additional Dimension Object* with `type` set to `temporal`.
      type: object
      required:
        - extent
      properties:
        values:
          description:>
            If the dimension consists of set of specific values they can be listed here. The dates and/or times MUST be strings compliant to [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601).
          type: array
          minItems: 1
          items:
            type: string
        extent:
          description:>
            Extent (lower and upper bounds) of the dimension as two-dimensional array. The dates and/or times MUST be strings compliant to [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601). `null` is allowed for open date ranges.
          type: array
          minItems: 2
          maxItems: 2
          items:
            type: string
          nullable: true
**Step**

- **Description**: The space between the temporal instances as [ISO 8601 duration](https://en.wikipedia.org/wiki/ISO_8601#Durations), e.g. `P1D`. Use `null` for irregularly spaced steps.
- **Type**: string
- **Nullable**: true

**Collection Dimension SRS**

- **Title**: Spatial reference system
- **Description**: The spatial reference system for the data, specified as [EPSG code](http://www.epsg-registry.org/), [WKT2 (ISO 19162) string](http://docs.opengeospatial.org/is/18-010r7/18-010r7.html), [PROJJSON object](https://proj.org/specifications/projjson.html) or [PROJ definition (deprecated)](https://proj.org/usage/quickstart.html). Defaults to EPSG code 4326.
- **Default**: 4326
- **OneOf**:
  - **Type**: number
  - **Title**: EPSG code
  - **Type**: string
  - **Title**: WKT2 or PROJ definition (deprecated)
  - **Type**: object
  - **Title**: PROJJSON

**Collection Dimension Extent Open**

- **Description**: If the dimension consists of ordinal [values](https://en.wikipedia.org/wiki/Level_of_measurement#Ordinal_scale), the extent (lower and upper bounds) of the values as two-dimensional array. Use `null` for open intervals.
- **Type**: array
- **MinItems**: 2
- **MaxItems**: 2
- **Items**:
  - **Type**: number
  - **Nullable**: true

**Collection Dimension Values**

- **Description**: A set of all potential values, especially useful for nominal [values](https://en.wikipedia.org/wiki/Level_of_measurement#Nominal_level).

**Important**: The order of the values MUST be exactly how the dimension values are also ordered in the data (cube). If the values specify band names, the values MUST be in the same order as they are in the corresponding band fields (i.e. `eo:bands` or `sar:bands`).
- **Type**: array
- **MinItems**: 1
- **Items**:
  - **OneOf**:
    - **Type**: number
    - **Type**: string

**Collection Dimension Step**

- **Description**: If the dimension consists of interval [values](https://en.wikipedia.org/wiki/Level_of_measurement#Interval_scale), the space between the values. Use `null` for irregularly spaced
steps.
type: number
nullable: true
collection_dimension_unit:
description: The unit of measurement for the data, preferably compliant to [UDUNITS-2](https://ncics.org/portfolio/other-resources/udunits2/) units (singular).
type: string
process_arguments:
title: Process Arguments
type: object
additionalProperties:
  $ref: '#/components/schemas/process_argument_value'
process_argument_value:
title: Process Argument Value
description: Arguments for a process. See the API documentation for more information.
nullable: true
anyOf:
  - type: object
    nullable: true
    title: Object (restricted)
    properties:
      from_parameter:
        not: {}
      from_node:
        not: {}
      process_graph:
        not: {}
  - type: string
    title: String
  - type: number
    title: Number (incl. integers)
  - type: boolean
    title: Boolean
  - type: array
    title: Array
    items:
      $ref: '#/components/schemas/process_argument_value'
  - $ref: '#/components/schemas/process_graph_with_metadata'
  - type: object
    title: Result Reference
    description: Data that is expected to be passed from another process.
    required:
      - from_node
    properties:
      from_node:
        description: The ID of the node that data is expected to come from.
        type: string
        additionalProperties: false
  - type: object
    title: Parameter Reference
    description: A parameter for a process graph. Data that is expected to be passed to a process graph either from the user directly or from the process that is executing the process graph.
    required:
      - from_parameter
    properties:
      from_parameter:
        description: The name of the parameter that data is expected to come from.
type: string
additionalProperties: false

process_graph:
title: Process Graph
description: A process graph defines a graph-like structure as a connected set of executable processes. Each key is a unique identifier (node ID) that is used to refer to the process in the graph.
description:

- type: object
  additionalProperties: x-additionalPropertiesName: Node ID
title: Process Node
description: Optional description about the process and its arguments.
description:

- type: string
  nullable: true
  arguments:
    $ref: '#/components/schemas/process_arguments'
  example:
  example:
dc:
  process_id: load_collection
  arguments:
    id: Sentinel-2
    spatial_extent:
      west: 16.1
      east: 16.6
      north: 48.6
      south: 47.2
    temporal_extent:
      - '2018-01-01'
      - '2018-02-01'
  bands:
    process_id: filter_bands
description: Filter and order the bands. The order is important for the following reduce operation.
description:
  arguments:
    data: from_node: dc
    bands:
      - B08
      - B04
      - B02
  evi:
    process_id: reduce
description: Compute the EVI. Formula: \(2.5 \times (NIR - RED) / (1 + NIR + 6 \times RED + -7.5 \times BLUE)\)

arguments:
data:
  from_node: bands
dimension: bands
reducer:
  process_graph:
nir:
    process_id: array_element
    arguments:
      data:
        from_parameter: data
        index: 0
red:
  process_id: array_element
  arguments:
    data:
      from_parameter: data
      index: 1
blue:
  process_id: array_element
  arguments:
    data:
      from_parameter: data
      index: 2
sub:
  process_id: subtract
  arguments:
    data:
      - from_node: nir
      - from_node: red
p1:
  process_id: product
  arguments:
    data:
      - 6
      - from_node: red
p2:
  process_id: product
  arguments:
    data:
      - -7.5
      - from_node: blue
sum:
  process_id: sum
  arguments:
    data:
      - 1
      - from_node: nir
      - from_node: p1
      - from_node: p2
div:
  process_id: divide
  arguments:
    data:
      - from_node: sub
      - from_node: sum
p3:
  process_id: product
  arguments:
    data:
mintime:
  process_id: reduce
  description: Compute a minimum time composite by reducing the temporal dimension
  arguments:
    data:
      from_node: evi
dimension: temporal
  reducer:
    process_graph:
      min:
        process_id: min
        arguments:
          data:
            from_parameter: data
  result: true
save:
  process_id: save_result
  arguments:
    data:
      from_node: mintime
  format: GTiff
  result: true
process:
title: Process
  type: object
  properties:
    id:
      $ref: '#/components/schemas/process_id'
    summary:
      $ref: '#/components/schemas/process_summary'
    description:
      $ref: '#/components/schemas/process_description'
    categories:
      $ref: '#/components/schemas/process_categories'
    parameters:
      $ref: '#/components/schemas/process_parameters'
    returns:
      $ref: '#/components/schemas/process_return_value'
    deprecated:
      $ref: '#/components/schemas/deprecated'
    experimental:
      $ref: '#/components/schemas/experimental'
    exceptions:
      $ref: '#/components/schemas/process_exceptions'
  examples:
    type: array
    description: Examples, may be used for unit tests.
    items:
      title: Process Example
      type: object
      required:
        - arguments
      properties:
        title:
          type: string
          description: A title for the example.
          description:
            $ref: '#/components/schemas/process_description'
        arguments:
returns:
  description: The return value which can be of any data type.
links:
  type: array
  description: |
    Links related to this process, e.g. additional external
documentation.

It is RECOMMENDED to provide links with the following `rel` (relation) types:

1. `latest-version`: If a process has been marked as deprecated, a
   link SHOULD point to the preferred version of the process. The relation types
   `predecessor-version` (link to older version) and `successor-version` (link to newer
   version) can also be used
   to show the relation between versions.

2. `example`: Links to examples of other processes that use this
   process.

3. `cite-as`: For all DOIs associated with the process, the
   respective DOI links SHOULD be added.

   For additional relation types see also the lists of
   [common relation types](#section/API-Principles/Web-Linking).

items:
  $ref: '#/components/schemas/link'
process_graph:
  $ref: '#/components/schemas/process_graph'
user_defined_process_meta:
  title: User-defined Process Metadata
  description: A user-defined process, may only contain metadata and no
  process graph.
  type: object
  required: - id
  properties:
    summary:
      type: string
      nullable: true
    description:
      type: string
      nullable: true
    parameters:
      type: array
      nullable: true
    returns:
      type: object
      nullable: true
  allOf:
    - $ref: '#/components/schemas/process'
process_graph_with_metadata:
  title: Process Graph with metadata
  description: A process graph, optionally enriched with process metadata.
  type: object
  required: - process_graph
  properties:
    id:
type: string  
nullable: true

summary:  
  type: string  
nullable: true
description:  
  type: string  
nullable: true

parameters:  
  type: array  
nullable: true
returns:  
  type: object  
nullable: true

allOf:  
  - $ref: '#/components/schemas/process'

process_namespace:  
  type: string  
nullable: true
default: null
example: null
description: The namespace the `process_id` is valid for.

The following options are predefined by the geodatcube API, but additional namespaces may be introduced by back-ends or in a future version of the API.

* `null` (default): Checks both user-defined and predefined processes, but prefers user-defined processes if both are available. This allows users to add missing predefined processes for portability, e.g. common processes from [processes.openeo.org](https://processes.openeo.org) that have a process graph included. It is RECOMMENDED to log the namespace selected by the back-end for debugging purposes.
* `backend`: Uses exclusively the predefined processes listed at `GET /processes`.
* `user`: Uses exclusively the user-defined processes listed at `GET /process_graphs`.

If multiple processes with the same identifier exist, Clients SHOULD inform the user that it’s recommended to select a namespace.

process_id:  
  type: string  
description: The identifier for the process. It MUST be unique across its namespace (e.g. predefined processes or user-defined processes).

Clients SHOULD warn the user if a user-defined process is added with the same identifier as one of the predefined process.

pattern: '^[a-zA-Z]+$'
example: ndvi

process_summary:  
  type: string  
description: A short summary of what the process does.

process_categories:  
  type: array  
description: A list of categories.
**process_return_value:**

- **type:** object
  - **title:** Process Return Value
  - **description:** Description of the data that is returned by this process.
  - **required:**
    - **schema**
  - **properties:**
    - **description:**
    - $ref: '#/components/schemas/process_description'
    - **schema:**
    - $ref: '#/components/schemas/process_schema'

**experimental:**

- **type:** boolean
  - **description:**
    - Declares that the specified entity is experimental, which means that it is likely to change or may produce unpredictable behaviour. Users should refrain from using it in production, but still feel encouraged to try it out and give feedback.
  - **default:** false

**deprecated:**

- **type:** boolean
  - **description:**
    - Declares that the specified entity is deprecated with the potential to be removed in any of the next versions. It should be transitioned out of usage as soon as possible and users should refrain from using it in new implementations.
  - **default:** false

**process_exceptions:**

- **type:** object
  - **title:** Process Exceptions
  - **description:**
    - Declares exceptions (errors) that might occur during execution of this process. This list is just for informative purposes and may be incomplete. This list MUST only contain exceptions that stop the execution of a process and MUST NOT contain warnings, notices or debugging messages. It is meant to primarily contain errors that have been caused by the user. It is RECOMMENDED that exceptions are referred to and explained in process or parameter descriptions.

The keys define the error code and MUST match the following pattern: `\w+`. The keys define the error code and MUST match the following pattern: `\w+`.

**additionalProperties:**

- **x-additionalPropertiesName:** Error Code
  - **title:** Process Exception
  - **type:** object
  - **required:**
    - **message**
  - **properties:**
    - **description:**
      - **type:** string
      - **format:** commonmark
      - **description:**
        - Detailed description to explain the error to client users and back-end developers. This should not be shown in the clients directly, but MAY be linked to in the errors `url` property.

        [CommonMark 0.29](http://commonmark.org/) syntax MAY be used for rich text representation.

  - **message:**
type: string
description: >-
   Explains the reason the server is rejecting the request. This
   message is intended to be displayed to the client user. For
   "4xx" error codes the message SHOULD explain shortly how the
   client needs to modify the request.

   The message MAY contain variables, which are enclosed by curly
   brackets. Example: `{variable_name}`
example: >-
   The value specified for the process argument '{argument}' in
   process '{process}' is invalid: {reason}

http:
  type: integer
description: >-
   HTTP Status Code, following the [error handling conventions in
   this API](#section/API-Principles/Error-Handling).
   Defaults to `400`.
default: 400

process_parameters:
  type: array
  description: |-
     A list of parameters.

The order in the array corresponds to the parameter order to
be used in clients that don't support named parameters.

**Note:** Specifying an empty array is different from (if allowed)
`null` or the property being absent.
An empty array means the process has no parameters.
`null` / property absent means that the parameters are unknown as
the user has not specified them. There could still be parameters in the
process graph, if one is specified.

items: $ref: '#/components/schemas/process_parameter'

base_parameter:
  type: object
  required:
    - name
    - description
  properties:
    name:
      type: string
      description: |-
         A unique name for the parameter.

      It is RECOMMENDED to use [snake case](https://en.wikipedia.org/
      wiki/snake_case) (e.g., `window_size` or `scale_factor`).
      pattern: '\w+$'
      description: $ref: '#/components/schemas/process_description'
    optional:
      type: boolean
      description: >>-
         Determines whether this parameter is optional to be specified even
when no default is specified.

         Clients SHOULD automatically set this parameter to `true`, if a
default value is specified.
         Back-ends SHOULD NOT fail, if a default value is specified and
this flag is missing.
      default: false
deprecated:
  $ref: '#/components/schemas/deprecated'

experimental:
  $ref: '#/components/schemas/experimental'

default:
  description: >-
    The default value for this parameter.
    Required parameters SHOULD NOT specify a default value. Optional
    parameters SHOULD always specify a default value.

parameter:
  title: Parameter
  type: object
  required:
  - schema
  properties:
    schema:
      $ref: '#/components/schemas/data_type_schema'
    allOf:
    - $ref: '#/components/schemas/base_parameter'

process_parameter:
  title: Process Parameter
  type: object
  required:
  - schema
  properties:
    schema:
      $ref: '#/components/schemas/process_schema'
    allOf:
    - $ref: '#/components/schemas/base_parameter'

batch_job:
  title: Batch Job
  description: >-
    The metadata of a batch jobs that has been submitted by the
    authenticated user.
  type: object
  required:
  - id
  - status
  - created
  properties:
    id:
      $ref: '#/components/schemas/job_id'
    title:
      $ref: '#/components/schemas/eo_title'
    description:
      $ref: '#/components/schemas/eo_description'
    process:
      $ref: '#/components/schemas/process_graph_with_metadata'
    status:
      type: string
      enum:
      - created
      - queued
      - running
      - canceled
      - finished
      - error
    description: |-
      The current status of a batch job.
      
      The following status changes can occur:
      * `POST /jobs`: The status is initialized as `created`
      * `POST /jobs/{job_id}/results`: The status is set to `queued`, if
processing doesn't start instantly.
  * Once the processing starts the status is set to `running`.
  * Once the data is available to download the status is set to

`finished`.
  * Whenever an error occurs during processing, the status MUST
be set to `error`.
  * `DELETE /jobs/{job_id}/results`: The status is set to `canceled` if
  the status was `running` beforehand and partial or preliminary

results
  are available to be downloaded. Otherwise the status is set to
  `created`.

example: running
default: created

progress:
type: number
description: `Indicates the process of a running batch job in percent.

Can also be set for a job which stopped due to an error or was
canceled by the user. In this case, the value indicates
the progress at which the job stopped. The Property may not be
available for the status codes `created` and `queued`.

Submitted and queued jobs only allow the value `0`

finished jobs only allow the value `100`.

minimum: 0
maximum: 100
example: 75.5

created:
$ref: `#/components/schemas/created`

updated:
$ref: `#/components/schemas/updated`

usage:
description: `Metrics about the resource usage of the batch job.

Back-ends are not expected to update the metrics while processing
data, so the metrics can only be available after the job has finished
or has stopped due to an error.

For usage metrics during processing, metrics can better be added
to the

logs (e.g. `GET /jobs/{job_id}/logs`) with the same schema.

allOf:
- $ref: `#/components/schemas/usage`

log_level:
$ref: `#/components/schemas/min_log_level_default`

links:
type: array
description: `Links related to this batch job, e.g. a links to
invoices, log files or results.

It is RECOMMENDED to provide links with the following `rel`
(relation) types:

1. `monitor`: If logs are available, a link to the [logs endpoint]
 (#tag/Batch-Jobs/operation/debug-job).
2. `result`: If batch job results are available, a link to the
 [results endpoint] (#tag/Batch-Jobs/operation/list-results).
The relation types `monitor` and `result` may occur for various batch job states:

1. `created`: When the batch job was executed before and has been reset to `created` after an [update](#tag/Batch-Jobs/operation/update-job) there could still be results and logs available until they get discarded by [queueing the batch job again](#tag/Batch-Jobs/operation/start-job).
2. `finished`: The full log and results are expected to be available.
3. `error` / `canceled`: Partial results and logs may be available.

For more relation types see the lists of [common relation types](#section/API-Principles/Web-Linking).

```json
example:
- rel: result
type: application/json
title: Batch Job Results
href: https://geodatacube.example/api/v1/jobs/123/logs
- rel: result
type: application/json
title: Batch Job Logs
href: https://geodatacube.example/api/v1/jobs/123/logs
```

```
job_id:
type: string
description: Per-backend unique identifier of the batch job, generated by the back-end during creation. MUST match the specified pattern.
pattern: '^[\w\-\.~]+$'
example: a3cca2b2aa1e3b5b
```

```
created:
type: string
format: date-time
description: Date and time of creation, formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.
example: '2017-01-01T09:32:12Z'
```

```
updated:
type: string
format: date-time
description: Date and time of the last status change, formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.
example: '2017-01-01T09:36:18Z'
```

```
description:
type: string
format: commonmark
description: Detailed description to explain the entity.
[CommonMark 0.29](http://commonmark.org/) syntax MAY be used for rich text representation.
```

```
object_title:
type: string
description: A human-readable short title to be displayed to users **in addition** to the names specified in the keys. This property is only for better user experience so that users can understand the names better.
```
Example titles could be `GeoTiff` for the key `GTiff` (for file formats) or `OGC Web Map Service` for the key `WMS` (for service types). The title MUST NOT be used in communication (e.g. in process graphs), although clients MAY translate the titles into the corresponding names.

- `eo_title`
  - description: A short description to easily distinguish entities.
  - type: string
  - nullable: true
  - example: NDVI based on Sentinel 2

- `eo_description`
  - type: string
  - format: commonmark
  - description: Detailed multi-line description to explain the entity.

[CommonMark 0.29](http://commonmark.org/) syntax MAY be used for rich text representation.

- `service`
  - title: Secondary Web Service
  - description: The metadata of a secondary web service that has been submitted by the authenticated user.
  - type: object
  - required:
    - id
    - enabled
    - type
    - url
  - properties:
    - id:
      - $ref: '#/components/schemas/service_id'
    - title:
      - $ref: '#/components/schemas/eo_title'
    - description:
      - $ref: '#/components/schemas/eo_description'
    - url:
      - type: string
        format: uri
        description: URL at which the secondary web service is accessible. Doesn't necessarily need to be located within the API.
        example: 'https://geodatacube.example/wms/wms-a3cca9'
      - $ref: '#/components/schemas/service_type'
    - enabled:
      - $ref: '#/components/schemas/service_enabled'
    - process:
      - type: string
        format: commonmark
        description: Detailed description to explain the entity.
configuration:
  $ref: '#/components/schemas/service_configuration'
attributes:
  title: Secondary Web Service Attributes
  type: object
  description: Additional attributes of the secondary web service, e.g. available layers for a WMS based on the bands in the underlying GeoTiff.
example:
  layers:
   - ndvi
   - evi
created:
  $ref: '#/components/schemas/created'
usage:
  description: Metrics about the resource usage of the secondary web service.
Back-ends are not expected to update the metrics in real-time. For detailed usage metrics for individual processing steps, metrics can be added to the logs (e.g. `GET /jobs/{job_id}/logs`) with the same schema.
allOf:
  - $ref: '#/components/schemas/usage'
log_level:
  $ref: '#/components/schemas/min_log_level_default'
service_type:
  description: Definition of the service type to access result data. All available service types can be retrieved via `GET /service_types`. Service types MUST be accepted in a *case insensitive* manner.
  type: string
example: wms
service_configuration:
  type: object
  title: Service Configuration
  description: Map of configuration settings, i.e. the setting names supported by the secondary web service combined with actual values. See `GET /service_types` for supported configuration settings. For example, this could specify the required version of the service, visualization details or any other service dependant configuration.
example:
  version: 1.3.0
service_enabled:
  type: boolean
  description: Describes whether a secondary web service is responding to requests (true) or not (false). Disabled services don't produce any costs.
service_id:
  type: string
  description: A per-backend unique identifier of the secondary web service, generated by the back-end during creation. MUST match the specified pattern.
  pattern: '^[\w\-\.\~]+$'
example: wms-a3cca9
resource_parameter:
  x-additionalPropertiesName: Parameter Name
  type: object
  title: Resource Parameter
Describes a parameter for various resources (e.g. file formats, service types).

The parameters are specified according to the [JSON Schema draft-07](http://json-schema.org/) specification.
See the chapter ['Schemas' in 'Defining Processes'](#section/Processes/Defining-Processes) for more information.

The following more complex JSON Schema keywords SHOULD NOT be used:
`if`, `then`, `else`, `readOnly`, `writeOnly`, `dependencies`, `minProperties`, `maxProperties`, `patternProperties`.

JSON Schemas SHOULD always be dereferenced (i.e. all `$refs` should be resolved). This allows clients to consume the schemas much better.
Clients are not expected to support dereferencing `$refs`.

Note: The specified schema is only a common subset of JSON Schema. Additional keywords MAY be used.

```json
required:
  - description
properties:
  description:
    type: string
    description: A brief description of the parameter according to [JSON Schema draft-07](https://json-schema.org/draft-07/json-schema-validation.html#rfc.section.10.1).
    required:
      type: boolean
      description: Determines whether this parameter is mandatory.
      default: false
    experimental:
      $ref: '#/components/schemas/experimental'
    default:
      description: The default value represents what would be assumed by the consumer of the input as the value of the parameter if none is provided. The value MUST conform to the defined type for the parameter defined at the same level. For example, if type is string, then default can be "foo" but cannot be 1. See [JSON Schema draft-07](https://json-schema.org/draft-07/json-schema-validation.html#rfc.section.10.2).
    allOf:
      - $ref: '#/components/schemas/process_json_schema'
error:
  title: General Error
  description: An error object declares additional information about a client-side or server-side error.
  See also: * [Error Handling](#section/API-Principles/Error-Handling) in the API in general.
  type: object
  required:
    - code
    - message
  properties:
    id:
      type: string
      description: A back-end MAY add a unique identifier to the error response to be able
A client could communicate this id to a back-end provider to get further information.

Example: 550e8400-e29b-11d4-a716-446655440000

code:

$ref: '#/components/schemas/log_code'
mmessage:

type: string
description: A message explaining what the client may need to change or what difficulties the server is facing.
example: Parameter 'sample' is missing.

links:

$ref: '#/components/schemas/log_links'

log_code:

type: string
description: The code is either one of the standardized error codes or a custom code, for example specified by a user in the `inspect` process.
example: SampleError

log_links:

description: Links related to this log entry / error, e.g. to a resource that provides further explanations.

For relation types see the lists of [common relation types](#section/API-Principles/Web-Linking).

type: array
items:

$ref: '#/components/schemas/link'
example:

- href: 'https://geodatacube.example/docs/errors/SampleError'
  rel: about

log_level:

description: The severity level of the log entry.

The order of the levels is as follows (from low to high severity): `debug`, `info`, `warning`, `error`.

The level `error` usually corresponds with critical issues that usually terminate the data processing.

type: string
enum:
- error
- warning
- info
- debug
example: error

min_log_level_default:

description: The minimum severity level for log entries that the back-end stores for the processing request.

The order of the levels is as follows (from low to high severity): `debug`, `info`, `warning`, `error`.
That means if `warning` is set, the back-end will only store log entries with the level `warning` and `error`.

The default minimum log level is `info`. Users need to specifically set this property to `debug` to get *all* log entries.
It is RECOMMENDED that users set the level at least to "warning" in production workflows.

```
type: string
enum:
- error
- warning
- info
- debug
default: info
example: warning
```

`min_log_level_update:`

```
description: |- Updates the minimum severity level for log entries that the back-end stores for the processing requests.
```

The back-end doesn't need to update existing log entries.

```
type: string
enum:
- error
- warning
- info
- debug
example: warning
```

`data_type_schema:`

```
title: Data Types
description: Either a single data type or a list of data types.
oneOf:
- $ref: '#/components/schemas/process_json_schema'
- title: Multiple data types
description: A list of data types this parameter supports, specified as JSON Schemas.
type: array
minItems: 1
uniqueItems: true
items:
  $ref: '#/components/schemas/process_json_schema'
```

`process_schema:`

```
title: Process Data types
description: Either a single data type or a list of data types for process parameter or process return values.
oneOf:
- $ref: '#/components/schemas/process_json_schema'
- title: Multiple data types
description: A list of data types supported, specified as JSON Schemas.
type: array
minItems: 1
uniqueItems: true
items:
  $ref: '#/components/schemas/process_json_schema'
```

`process_json_schema:`

```
type: object
title: Single Data Type
description: |- Specifies a data type supported by a parameter or return value.
```

The data types are specified according to the [JSON Schema draft-07]
(http://json-schema.org/) specification.

See the chapter ['Schemas' in 'Defining Processes'](#section/Processes/Defining-Processes) for more information.

JSON Schemas SHOULD NOT contain `default`, `anyOf`, `oneOf`, `allOf` or `not` at the top-level of the schema.
Instead specify each data type in a separate array element.

The following more complex JSON Schema keywords SHOULD NOT be used:
`if`, `then`, `else`, `readOnly`, `writeOnly`, `dependencies`, `minProperties`, `maxProperties`, `patternProperties`.

JSON Schemas SHOULD always be dereferenced (i.e. all `$refs` should be resolved). This allows clients to consume the schemas much better. Clients are not expected to support dereferencing `$refs`.

Note: The specified schema is only a common subset of JSON Schema. Additional keywords MAY be used.

```
properties:
  subtype:
    type: string
    description: The allowed sub data type for a value. See the chapter on [subtypes](#section/Processes/Defining-Processes) for more information.
  deprecated:
    $ref: '#/components/schemas/deprecated'
  allOf:
    - $ref: '#/components/schemas/json_schema'
  oneOf:
    - title: Generic
      - $ref: '#/components/schemas/process_graph_json_schema'
    - $ref: '#/components/schemas/datacube_json_schema'

process_graph_json_schema:
  title: Process Graph
  type: object
  properties:
    subtype:
      type: string
      enum:
        - process-graph
    parameters:
      type: array
      description: A list of parameters passed to the child process graph.
      items:
        $ref: '#/components/schemas/parameter'
  returns:
    type: object
    title: Process Graph Return Value
    description: Description of the data that is returned by the child process graph.
    required:
      - schema
    properties:
      description:
        $ref: '#/components/schemas/process_description'
      schema:
        $ref: '#/components/schemas/data_type_schema'
      allOf:
        - $ref: '#/components/schemas/process_json_schema'

datacube_json_schema:
  title: Datacube
  properties:
    subtype:
      type: string
```
enum:
  - datacube
dimensions:
title: Datacube constraints
description:
  Allows to specify requirements the data cube has to fulfill. Right now, it only allows to specify the dimension types and adds for specific dimension types:
  * axes for `spatial` dimensions in raster datacubes
  * geometry types for `geometry` dimensions in vector datacubes
type: array
items:
type: object
required:
  - type
oneOf:
  - title: Spatial (raster)
    properties:
      type:
        type: string
        enum:
        - spatial
      axis:
        type: array
        minItems: 1
        items:
          $ref: '#/components/schemas/dimension_axis_xyz'
  - title: Spatial (vector)
    properties:
      type:
        type: string
        enum:
        - geometry
      geometry_type:
        type: array
        minItems: 1
        items:
          $ref: '#/components/schemas/geometry_type'
  - title: Other
    properties:
      type:
        type: string
        enum:
        - bands
        - temporal
        - other

json_schema:
type: object
title: JSON Schema
description:
  A JSON Schema compliant to [JSON Schema draft-07](https://json-schema.org/draft-07/json-schema-validation.html) or later.

  JSON Schemas SHOULD always be dereferenced (i.e. all `$refs` should be resolved). This allows clients to consume the schemas much better. Clients are not expected to support dereferencing `$refs`.

  Note: The specified schema in the OpenAPI document is only a common subset of JSON Schema. Additional keywords from the JSON Schema specification MAY be used.

properties:
$schema:
description: |-  
The JSON Schema version. If not given in the context of this API,  
defaults to `draft-07`.

You may need to add the default value for `$schema` property  
explicitly to the JSON Schema  
object before passing it to a JSON Schema validator.

type: string  
format: uri  
default: http://json-schema.org/draft-07/schema#

$id:  
description: ID of your JSON Schema.  
type: string  
format: uri  
type:  
description: |-

The allowed basic data type(s) for a value.

If this property is not present, all data types are allowed.

oneOf:
- $ref: '#/components/schemas/json_schema_type'  
- type: array  
  minItems: 1  
  uniqueItems: true  
  items:  
    $ref: '#/components/schemas/json_schema_type'

pattern:  
type: "string"  
format: "regex"  
description: The regular expression a string value must match  
against.

enum:  
type: array  
items: {}  
description: An exclusive list of allowed values.

minimum:  
type: number  
description: The minimum value (inclusive) allowed for a numerical  
value.

maximum:  
type: number  
description: The maximum value (inclusive) allowed for a numerical  
value.

minItems:  
type: number  
minimum: 0  
default: 0  
description: The minimum number of items required in an array.

maxItems:  
type: number  
minimum: 0  
description: The maximum number of items required in an array.

items:  
description: Specifies schemas for the items in an array.

anyOf:  
- type: array  
  minItems: 1  
  items:  
    $ref: '#/components/schemas/json_schema_type'  
    - $ref: '#/components/schemas/json_schema_type'

additionalProperties:  
description: >-
You can add any other property supported by the JSON Schema version that is given through the property `$schema`, so either [draft-07](https://json-schema.org/draft-07/json-schema-validation.html) or any later version.

```json
json_schema_type:
  type: string
enum:
  - array
  - boolean
  - integer
  - 'null'
  - number
  - object
  - string

geometry_type:
  title: Geometry type
  type: string
enum:
  - Point
  - MultiPoint
  - LineString
  - MultiLineString
  - Polygon
  - MultiPolygon
  - GeometryCollection

GeoJsonPoint3D:
  type: array
  description: Point in 3D space
  minItems: 2
  maxItems: 3
  items:
    type: number

GeoJsonPoint:
  type: object
  title: GeoJSON Point
  required:
    - type
    - coordinates
  properties:
    type:
      type: string
      enum:
        - Point
    coordinates:
      $ref: '#/components/schemas/GeoJsonPoint3D'

GeoJsonFeatureCollection:
  type: object
  required:
    - type
    - features
  properties:
    type:
      type: string
      enum:
        - FeatureCollection
    features:
      type: array
      items:
        $ref: '#/components/schemas/GeoJsonFeature'

GeoJsonFeature:
  type: object
  required:
    - type
- geometry
  - properties
    type:
      type: string
    enum:
      - Feature
    geometry:
      $ref: '#/components/schemas/GeoJsonGeometry'
    properties:
      type: object
      nullable: true

GeoJsonGeometry:
  title: GeoJSON Geometry
  type: object
  required:
  - type
  properties:
    type:
      $ref: '#/components/schemas/geometry_type'
    discriminator:
      propertyName: type
    mapping:
      Point: '#/components/schemas/GeoJsonPoint'
      LineString: '#/components/schemas/GeoJsonLineString'
      Polygon: '#/components/schemas/GeoJsonPolygon'
      MultiPoint: '#/components/schemas/GeoJsonMultiPoint'
      MultiLineString: '#/components/schemas/GeoJsonMultiLineString'
      MultiPolygon: '#/components/schemas/GeoJsonMultiPolygon'
      GeometryCollection: '#/components/schemas/GeoJsonGeometryCollection'

GeoJsonLineString:
  allOf:
  - $ref: '#/components/schemas/GeoJsonGeometry'
  - type: object
    title: GeoJSON LineString
    required:
      - coordinates
    properties:
      coordinates:
        type: array
        items:
          $ref: '#/components/schemas/GeoJsonPoint3D'

GeoJsonPolygon:
  allOf:
  - $ref: '#/components/schemas/GeoJsonGeometry'
  - type: object
    title: GeoJSON Polygon
    required:
      - coordinates
    properties:
      coordinates:
        type: array
        items:
          type: array
          items:
            $ref: '#/components/schemas/GeoJsonPoint3D'

GeoJsonMultiPoint:
  allOf:
  - $ref: '#/components/schemas/GeoJsonGeometry'
  - type: object
    title: GeoJSON MultiPoint
    required:
      - coordinates
properties:
  coordinates:
    type: array
    items:
      $ref: '#/components/schemas/GeoJsonPoint3D'
GeoJsonMultiLineString:
  allOf:
    - $ref: '#/components/schemas/GeoJsonGeometry'
    - type: object
      title: GeoJSON MultiLineString
      required:
        - coordinates
      properties:
        coordinates:
          type: array
          items:
            type: array
            items:
              $ref: '#/components/schemas/GeoJsonPoint3D'
GeoJsonMultiPolygon:
  allOf:
    - $ref: '#/components/schemas/GeoJsonGeometry'
    - type: object
      title: GeoJSON MultiPolygon
      required:
        - coordinates
      properties:
        coordinates:
          type: array
          items:
            type: array
            items:
              type: array
              items:
                $ref: '#/components/schemas/GeoJsonPoint3D'
GeoJsonGeometryCollection:
  allOf:
    - $ref: '#/components/schemas/GeoJsonGeometry'
    - type: object
      title: GeoJSON GeometryCollection
      required:
        - geometries
      properties:
        geometries:
          type: array
          items:
            $ref: '#/components/schemas/GeoJsonGeometry'
log_entry:
  title: Log Entry
  description: >-
    An log message that communicates information about the processed data.
  type: object
  required:
    - id
    - level
    - message
  properties:
    id:
      type: string
      description: >-
        An unique identifier for the log message, could simply be an incrementing number.
      example: "1"
code:  
$ref: '#/components/schemas/log_code'

level:  
$ref: '#/components/schemas/log_level'

message:  
type: string  
description: >-
  A concise message explaining the log entry.

  Messages do *not* explicitly support [CommonMark 0.29](http://commonmark.org/) syntax as other descriptive fields in the geodatacube API do, but the messages MAY contain line breaks or indentation.

  It is NOT RECOMMENDED to add stacktraces to the `message`.

  example: >-
  Can't load the UDF file from the URL `https://geodatacube.example/invalid/file.txt`.
  Server responded with error 404.

time:  
type: string  
format: date-time  
title: Date and Time  
description: >-
  The date and time the event happened, in UTC. Formatted as a [RFC 3339](https://www.rfc-editor.org/rfc/rfc3339.html) date-time.

data:  
description: |-
  Data of any type. It is the back-ends task to decide how to best present passed data to a user.

  For example, a datacube passed to the `inspect` SHOULD return the metadata similar to the collection metadata, including `cube:dimensions`.

  There are implementation guidelines available for the `inspect` process.

path:  
description: |-
  Describes where the log entry originates from.

  The first element of the array is the process that has triggered the log entry, the second element is the parent of the process that has triggered the log entry, etc. This pattern is followed until the root of the process graph.

type: array  
items:  
type: object  
required:  
  - node_id  
properties:  
node_id:  
type: string  
description: The id of the node the log entry originates from.

  example: runudf1  
process_id:  
$ref: '#/components/schemas/process_id'

namespace:  
$ref: '#/components/schemas/process_namespace'

parameter:  
type: string  
description: >-
  If applicable, the name of the parameter the log entry corresponds to.
usage:
  $ref: '#/components/schemas/usage'
links:
  $ref: '#/components/schemas/log_links'
usage:
  title: Resource usage metrics
  type: object
  properties:
    cpu:
      description: |
        Specifies the CPU usage, usually in a unit such as `cpu-seconds`
      allOf:
        - $ref: '#/components/schemas/usage_metric'
    memory:
      description: |
        Specifies the memory usage, usually in a unit such as `mb-seconds`
        or `gb-hours`
      allOf:
        - $ref: '#/components/schemas/usage_metric'
    duration:
      description: |
        Specifies the wall time, usually in a unit such as `seconds`, `minutes` or `hours`
      allOf:
        - $ref: '#/components/schemas/usage_metric'
    network:
      description: |
        Specifies the network transfer usage (incoming and outgoing), usually in a unit such as `b` (bytes), `kb` (kilobytes), `mb` (megabytes) or `gb` (gigabytes).
      allOf:
        - $ref: '#/components/schemas/usage_metric'
    disk:
      description: |
        Specifies the amount of input (read) and output (write) operations on the storage such as disks, usually in a unit such as `b` (bytes), `kb` (kilobytes), `mb` (megabytes) or `gb` (gigabytes).
      allOf:
        - $ref: '#/components/schemas/usage_metric'
    storage:
      description: |
        Specifies the usage of storage space, usually in a unit such as `b` (bytes), `kb` (kilobytes), `mb` (megabytes) or `gb` (gigabytes).
      allOf:
        - $ref: '#/components/schemas/usage_metric'
  additionalProperties:
    description: |
      Additional metrics.
    allOf:
      - $ref: '#/components/schemas/usage_metric'
example:
  cpu:
    value: 40668
    unit: cpu-seconds
  duration:
    value: 2611
    unit: seconds
  memory:
    value: 108138811
    unit: mb-seconds
network:
  value: 0
  unit: kb
storage:
  value: 55
  unit: mb
usage_metric:
  type: object
  required:
    - value
    - unit
  properties:
    value:
      type: number
      minimum: 0
    unit:
      type: string
responses:
  logs:
    description: Lists the requested log entries.
    content:
      application/json:
        schema:
          title: Log Entries
          type: object
          required:
            - logs
            - links
          properties:
            level:
              description: The minimum severity level for log entries that the back-end returns.
              type: string
              enum:
                - error
                - warning
                - info
                - debug
              default: debug
            logs:
              description: A chronological list of logs.
              type: array
              items:
                $ref: '#/components/schemas/log_entry'
            links:
              $ref: '#/components/schemas/links_pagination'
  client_error:
    description: The request can't be fulfilled due to an error on client-side, i.e. the request is invalid. The client SHOULD NOT repeat the request without modifications.
The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in [RFC 7231](https://www.rfc-editor.org/rfc/rfc7231.html#section-6.6). This request usually does not respond with HTTP status codes 401 and 403 due to missing authorization. HTTP status code 404 SHOULD be used if the value of a path parameter is invalid.

See also:
* [Error Handling](#section/API-Principles/Error-Handling) in the API in general.

```json
content:
  application/json:
    schema:
      $ref: '#/components/schemas/error'
client_error_auth:
  description: |
    The request can't be fulfilled due to an error on client-side, i.e. the request is invalid. The client SHOULD NOT repeat the request without modifications.
```

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in [RFC 7231](https://www.rfc-editor.org/rfc/rfc7231.html#section-6.6). This request MUST respond with HTTP status codes 401 if authorization is required or 403 if the authorization failed or access is forbidden in general to the authenticated user. HTTP status code 404 SHOULD be used if the value of a path parameter is invalid.

See also:
* [Error Handling](#section/API-Principles/Error-Handling) in the API in general.

```json
content:
  application/json:
    schema:
      $ref: '#/components/schemas/error'
server_error:
  description: |
    The request can't be fulfilled due to an error at the back-end. The error is never the client's fault and therefore it is reasonable for the client to retry the exact same request that triggered this response.
```

The response body SHOULD contain a JSON error object. MUST be any HTTP status code specified in [RFC 7231](https://www.rfc-editor.org/rfc/rfc7231.html#section-6.6).

See also:
* [Error Handling](#section/API-Principles/Error-Handling) in the API in general.

```json
content:
  application/json:
    schema:
      $ref: '#/components/schemas/error'
parameters:
  ogc_processID:
    name: processID
    description: ID of the OGC process
    in: path
    required: true
```
The format of the response. If no value is provided, the accept header is used to determine the format. Accepted values are 'json' or 'html'.

Retrieve only part of the data by slicing or trimming along one or more axis.

For trimming: `{axisAbbrev}({low}:{high})` (preserves dimensionality)
   An asterisk (`*`) can be used instead of `{low}` or `{high}` to indicate the minimum/maximum value.

For slicing: `{axisAbbrev}({value})` (reduces dimensionality)

reproject the output to the given crs

crs for the specified subset

crs for the specified bbox
scale-factor:
  name: scale-factor
  in: query
  description: `- For each axis, the returned coverage will contain the number of original sampled values, divided by the scale-factor.
  required: false
  schema:
    type: number

scale-axes:
  name: scale-axes
  in: query
  description: `- Returns a coverage re-scaled so as to contain `{number}` times less sample values along the corresponding axisName axis, and all original values along the dimensions of unspecified axes.
  ScalingSpec: "scale-axes"=axisName({number})[,
    axisName({number})]*
  axisName: {NCName}
  required: false
  schema:
    type: string

scale-size:
  name: scale-size
  in: query
  description: `- When `scale-size` is used, the returned coverage will contain exactly the specified number of sample values along each axis which is specified, and the original number of sample values for unspecified axes.
  ScalingSpec: "scale-size"=axisName({number})[,
    axisName({number})]*
  axisName: {text}
  required: false
  schema:
    type: string

properties:
  name: properties
  in: query
  description: `- Select specific data record fields (measured/observed properties) to be returned.
  RangeSubsetSpec: "properties"=field[,fieldName]*
  field: {fieldName}|{fieldIndex}|"*"
fieldName: {text}
fieldIndex: {number}

Where:

{number} is an integer number, and
{text} is some general ASCII text.

The field name must be one of the id defined in the RangeType DataRecord fields.
The field index must be an integer between 0 and the number of fields
- 1 defined in the RangeType DataRecord fields.
An asterisk indicates to also include subsequent fields.

required: false
schema:
  type: string
f-coverage:
  name: f
description: The optional f parameter indicates the output format which
the server shall provide as part of the response document. It has preference over
the HTTP Accept header.
explode: false
in: query
required: false
schema:
  type: string
  enum:
  - png
  - geotiff
  - netcdf
  - json
  - covjson
  - html
style: form
f-rangeset:
  name: f
description: The optional f parameter indicates the output format which
the server shall provide as part of the response document. It has preference over
the HTTP Accept header.
explode: false
in: query
required: false
schema:
  default: json
  enum:
  - json
  - html
  type: string
style: form
f-domainset:
  name: f
description: The optional f parameter indicates the output format which
the server shall provide as part of the response document. It has preference over
the HTTP Accept header.
explode: false
in: query
required: false
required: false
The optional f parameter indicates the output format which the server shall provide as part of the response document. It has preference over the HTTP Accept header.

Pagination is OPTIONAL: back-ends or clients may not support it. Therefore it MUST be implemented in a way that clients not supporting pagination get all resources regardless. Back-ends not supporting pagination MUST return all resources.

If the response is paginated, the `links` array MUST be used to communicate the links for browsing the pagination with predefined `rel` types. See the `links` array schema for supported `rel` types. Back-end implementations can, unless specified otherwise, use all kind of pagination techniques, depending on what is supported best by their infrastructure: page-based, offset-based, token-based or something else. The clients SHOULD use whatever is specified in the links with the corresponding `rel` types.

The last identifier (property `id` of a log entry) the client has received. If provided, the back-ends only sends the entries that
occurred after the specified identifier. If not provided or empty, start with
the first entry.

in: query
  allowEmptyValue: true
  example: log1234
  schema:
    type: string

log_level:
  name: level
description: |-nThe minimum severity level for log entries that the back-end returns.

The order of the levels is as follows (from low to high severity):
  `debug`, `info`, `warning`, `error`.
That means if `warning` is set, the back-end will only return log
entries with the level `warning` and `error`.

The default minimum log level is `debug`, which returns all log levels.

in: query
  allowEmptyValue: true
  example: error
  schema:
    type: string
    enum:
      - error
      - warning
      - info
      - debug
    default: info

service_id:
  name: service_id
  in: path
description: Identifier of the secondary web service.
  required: true
  schema:
    $ref: '#/components/schemas/service_id'

job_id:
  name: job_id
  in: path
description: Identifier of the batch job.
  required: true
  schema:
    $ref: '#/components/schemas/job_id'

collection_id:
  name: collection_id
  in: path
description: Collection identifier
  required: true
  schema:
    $ref: '#/components/schemas/collection_id'

bbox:
  name: bbox
  in: query
description: |-Only features that have a geometry that intersects the bounding box
are selected.
The bounding box is provided as four or six numbers, depending on
whether the
coordinate reference system includes a vertical axis (height or depth):

* Lower left corner, coordinate axis 1
* Lower left corner, coordinate axis 2
* Minimum value, coordinate axis 3 (optional)
The coordinate reference system of the values is WGS 84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84).

For WGS 84 longitude/latitude the values are in most cases the sequence of minimum longitude, minimum latitude, maximum longitude and maximum latitude. However, in cases where the box spans the antimeridian the first value (west-most box edge) is larger than the third value (east-most box edge).

If the vertical axis is included, the third and the sixth number are the bottom and the top of the 3-dimensional bounding box.

If a feature has multiple spatial geometry properties, it is the decision of the server whether only a single spatial geometry property is used to determine the extent or all relevant geometries.

required: false

schema:
  type: array
  oneOf:
    - minItems: 4
      maxItems: 4
    - minItems: 6
      maxItems: 6
  items:
    type: number
    style: form
  explode: false

datetime:
  name: datetime
  in: query
  description: |-
    Either a date-time or an interval, open or closed. Date and time expressions adhere to RFC 3339. Open intervals are expressed using double-dots.

Examples:

* A date-time: "2018-02-12T23:20:50Z"
* A closed interval: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z"
* Open intervals: "2018-02-12T00:00:00Z/.." or "/../2018-03-18T12:31:12Z"

Only features that have a temporal property that intersects the value of `datetime` are selected.

If a feature has multiple temporal properties, it is the decision of the server whether only a single temporal property is used to determine the extent or all relevant temporal properties.

required: false

schema:
  type: string
  style: form
```
explode: false
feature_id:
  name: feature_id
  in: path
  description: local identifier of a feature
  required: true
  schema:
    type: string
examples:
evi_user_defined_process:
  description: A user-defined process that computes the EVI.
  value:
    id: evi
    summary: Enhanced Vegetation Index
    description: Computes the Enhanced Vegetation Index (EVI).
    It is computed with the following formula: `2.5 * (NIR - RED) / (1 + NIR + 6*RED + -7.5*BLUE)`.
parameters:
- name: red
  description: Value from the red band.
  schema:
    type: number
- name: blue
  description: Value from the blue band.
  schema:
    type: number
- name: nir
  description: Value from the near infrared band.
  schema:
    type: number
returns:
  description: Computed EVI.
  schema:
    type: number
process_graph:
  sub:
    process_id: subtract
    arguments:
      x:
        from_parameter: nir
      y:
        from_parameter: red
    p1:
      process_id: multiply
      arguments:
        x: 6
        y:
          from_parameter: red
    p2:
      process_id: multiply
      arguments:
        x: -7.5
        y:
          from_parameter: blue
    sum:
      process_id: sum
      arguments:
        data:
          - 1
          - from_parameter: nir
          - from_node: p1
          - from_node: p2
```
div:
  process_id: divide
  arguments:
    x:
      from_node: sub
    y:
      from_node: sum

p3:
  process_id: multiply
  arguments:
    x: 2.5
    y:
      from_node: div
  result: true

securitySchemes:
  Bearer:
    type: http
    scheme: bearer
    bearerFormat: >-
    The Bearer Token MUST consist of the authentication method, a provider ID (if available) and the token itself. All separated by a forward slash `\/`. Examples (replace `\`TOKEN` with the actual access token): (1) Basic authentication (no provider ID available): `basic\//TOKEN` (2) OpenID Connect (provider ID is `ms`): `oidc/ms/TOKEN`. For OpenID Connect, the provider ID corresponds to the value specified for `id` for each provider in `GET /credentials/oidc`.

Basic:
  type: http
  scheme: basic

Figure B.1