Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/is/ogcapi-routes-1/1.0

Internal reference number of this OGC® document: 21-000

Version: 1.0.0-SNAPSHOT

Category: OGC® Implementation Specification

Editor: Clemens Portele, <other Editors>

OGC API - Routes - Part 1: Core

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® Standard
Document subtype:	if applicable
Document stage:	Draft
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

i. Abstract

TODO

ii. Keywords

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

OGC, Routing, Routing API

iii. Preface

TODO: Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work.

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.

iv. Submitting organizations

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

Organization name(s)

TODO

v. Submitters

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

Name Affiliation

TODO

Chapter 1. Scope

TODO: Insert Scope text here. Give the subject of the document and the aspects of that scope covered by the document.

Chapter 2. Conformance

TODO

This Standard defines XXXX.

Requirements for N standardization target types are considered:

- AAAA
- BBBB

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

In order to conform to this OGC® Standard, a software implementation shall choose to implement:

- Any one of the conformance levels specified in Annex A (normative).
- Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the Standard(s) identified.

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

TODO: Route Exchange Model, OGC API Common Core, ...

Chapter 4. Terms and Definitions

This document used the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the 'ModSpec'. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

TODO

term name

text of the definition

term name

text of the definition

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document.

TODO: add information about the use of OpenAPI 3.0, link relation types, etc.

5.1. Identifiers

The normative provisions in this Standard are denoted by the URI

http://www.opengis.net/spec/ogcapi-routes-1/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

Chapter 6. Overview

TODO: add more context

6.1. Requirements for Web-based Routing

This section describes the requirements that this standard supports.

The bare minimum requirement for a Routing API is to asynchronously compute a route based on a start and end point. The resulting route is encoded using the Route Exchange Model. This requirement enables implementations to work in all types of connectivity situations, including Denied, Disrupted, Intermittent, and Limited (DDIL) bandwidth communications networks.

Besides this basic requirement, additional optional requirements to the API are the following:

- Deletion of a computed route
- Support to call a webhook once the route calculation is completed
- Synchronous route computation
- Request parts of the Route Exchange Model, e.g. only the overview

Additionally, an API may support the following optional input parameters:

- Capability to add additional intermediate waypoints
- Apply a height restriction to the route computation
- Apply a load restriction to the route computation
- Ability to define obstacles that have to be avoided
- Specify departure or arrival times
- Select a specific routing engine
- Select the routing algorithm to use
- Select the source dataset to use

Chapter 7. Routing API

7.1. Overview

This clause specifies the Routing API as a Web API for routing based on the OGC API principles and OGC API - Common as well as the Route Exchange Model. It consists of the following requirements/conformance classes, each with a unique Uniform Resource Identifier (URI):

- Core: minimal routing capability (asynchronous routing based on start/end point)
- Delete route: Cancel/delete routes
- Callback: Support for call to a webhook after the route is computed
- Result set: request parts of the Route Exchange Model, e.g., in DDIL situations
- Synchronous execution: return the route in the response to the routing request
- Intermediate waypoints: Pass through additional points along the route
- Height restriction: Consider height restrictions
- Load restriction: Consider load restrictions
- Obstacles: Avoid obstacles
- Temporal constraints: Specify departure/arrival time
- Routing engine: Select a specific routing engine
- Routing algorithm: Select the algorithm to use
- Source dataset: Select the network dataset to use

This API supports the resources and operations listed in Table 1.

Table 1. Overview of resources and applicable HTTP methods

Resource	Path	HTTP method	Document reference
Landing page	`/`	GET	API landing page
Conformance declaration	`/conformance`	GET	Declaration of conformance classes
Routes	`/routes`	GET	Fetch routes
		POST	Compute a new route
Route	`/routes/{routeId}`	GET	Fetch a route
		DELETE	Delete a route
Route definition	`/routes/{routeId}/definition`	GET	Fetch the definition of a route

All resources are available in the 'Core' requirements class, i.e. all routing APIs will support them, with the exception of the operation to delete a route which is specified in the 'Delete route' requirements class.

7.2. Requirements Class "Core"

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/core	
Target type	Web API
Dependency	OGC API - Common (Core)
Dependency	GeoJSON
Dependency	http://www.opengis.net/spec/rem/1.0/req/rem-full

7.2.1. OGC API Common (Core)

At the time of writing, the OGC API - Common draft specification is not yet an approved standard. A draft is in development, based on the generic concepts of the OGC API - Features - Part 1: Core standard (OGC 17-069r3), but it is in an early stage of the standardization process. In order to avoid duplicating content this document does not copy the basic requirements, recommendations and permissions from OGC API Common/Features. The following normative statements are by reference part of this requirements class:

- Landing page
  - Requirement /req/core/root-op
  - Requirement /req/core/root-success
    - Change: No `data` link to `/collections` is required, but the `data` link has to point to the `/routes` resource
- API definition
  - Requirement /req/core/api-definition-op
  - Permission /per/core/api-definition-uri
  - Requirement /req/core/api-definition-success
  - Recommendation /rec/core/api-definition-oas
- Conformance declaration
  - Requirement /req/core/conformance-op
  - Requirement /req/core/conformance-success
- Web API
  - Requirement /req/core/http
  - Recommendation /rec/core/head
  - Permission /per/core/additional-status-codes
  - Requirement /req/core/query-param-unknown
  - Requirement /req/core/query-param-invalid
  - Recommendation /rec/core/etag

    - Recommendation /rec/core/cross-origin
    - Recommendation /rec/core/link-header
- Geometries
    - Requirement /req/core/crs84

The recommendation /rec/core/string-i18n is mainly implemented by the Content-Language header in the response to requests returning a route.

7.2.2. API landing page

The following is an example of the landing page of a routing API.

Example 1. Landing page


```
{
  "links": [
    {
      "href": "https://example.org/api/routing/v1",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": "https://example.org/api/routing/v1/api",
      "rel": "service-desc",
      "type": "application/vnd.oai.openapi+json;version=3.0",
      "title": "the API definition in OpenAPI JSON"
    },
    {
      "href": "https://example.org/api/routing/v1/api.html",
      "rel": "service-doc",
      "type": "text/html",
      "title": "the API documentation in HTML"
    },
    {
      "href": "https://example.org/api/routing/v1/conformance",
      "rel": "conformance",
      "type": "application/json",
      "title": "list of conformance classes implemented by this API"
    },
    {
      "href": "https://example.org/api/routing/v1/routes",
      "rel": "data",
      "type": "application/json",
      "title": "the routes"
    }
  ]
}
```


7.2.3. Declaration of conformance classes

The following is an example of the conformance declaration of a routing API that implements all requirements classes.

Some requirements classes support options and parsing the OpenAPI definition may be unnecessarily costly for clients to determine the options. The conformance declaration, therefore, is extended to support stating the options implemented.

Example 2. Conformance declaration


```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-processes-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/intermediate-waypoints",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/max-height",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/max-weight",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/obstacles",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-engine",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-algorithm",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/source-dataset",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/time",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/callback",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/result-set",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/sync-mode",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/delete-route"
  ],
  "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/core": {
    "values": [
      "fastest",
      "shortest"
    ]
  },
  "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-engine": {
    "values": [
      "Skymantics",
      "Ecere",
      "HERE"
    ]
  },
  "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-algorithm": {
    "values": [
      "Dikjstra",
      "Floyd Marshall",
      "A*"
    ]
  },
  "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/source-dataset": {
    "values": [
      "NSG",
      "OSM",
      "HERE"
    ]
  }
}
```

7.2.4. Geometries

All geometries used in the API are GeoJSON geometries. This includes the waypoints in the route definition and the geometries of all features in the route exchange model (overview, start, end, segments).

All geometries use coordinates based on the World Geodetic System 1984 (WGS 84) datum i.e. the coordinate reference system used by Global Positioning System (GPS). In GeoJSON, a coordinate is an array of numbers. The first two elements are longitude and latitude, or easting and northing, precisely in that order and using decimal numbers. Elevation may be included as an optional third element.

7.2.5. Routes

Fetch routes

This operation returns a list of routes that are currently available.

Requirement 1	**/req/core/routes-op**
A	The server SHALL support the HTTP GET operation at the path `/routes`.

Requirement 2	**/req/core/routes-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

B	The content of that response SHALL be based upon the following OpenAPI 3.0 schema:
	
	`type: object`
	`properties:`
	`  links:`
	`  type: array`
	`  items:`
	`  type: object`
	`  required:`
	`  - href`
	`  properties:`
	`  href:`
	`  type: string`
	`  rel:`
	`  type: string`
	`  type:`
	`  type: string`
	`  hreflang:`
	`  type: string`
	`  title:`
	`  type: string`

C	The links SHALL include a link (link relation `item`) to each route currently on the server.
D	If a route has a name, the name SHALL be used in the link title.

Example 3. Routes


```
{
  "links": [
    {
      "href": "https://example.org/api/routing/v1/routes",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": "https://example.org/api/routing/v1/routes/5hsb32",
      "rel": "item",
      "type": "application/geo+json",
      "title": "Lincoln Memorial to hotel"
    },
    {
      "href": "https://example.org/api/routing/v1/routes/9fg3dh",
      "rel": "item",
      "type": "application/geo+json",
      "title": "Lafayette Square to Zoo"
    },
    {
      "href": "https://example.org/api/routing/v1/routes/j6gdg3",
      "rel": "item",
      "type": "application/geo+json",
      "title": "DCA to hotel"
    }
  ]
}
```


Compute a new route

This operation creates a new route. The payload of the request specifies the definition of the new route.

The core requirements class supports a minimum route definition by two `waypoints`, the start and end point of the route.

In addition, clients can select 'fastest' or 'shortest' as the routing `preference`. The default value is 'fastest'.

An optional `name` for the route may be provided. The name will be used as the title in links to the route and is also included in the route itself.

Requirement 3	**/req/core/compute-route-op**

A	The server SHALL support the HTTP POST operation at the path `/routes`.

B	The server SHALL accept a route definition in the content of the request based upon the following OpenAPI 3.0 schema:
	
	`type: object`
	`required:`
	`  - waypoints`
	`properties:`
	`  name:`
	`  type: string`
	`  waypoints:`
	`  type: object`
	`  required:`
	`  - type`
	`  - coordinates`
	`  properties:`
	`  type:`
	`  type: string`
	`  enum:`
	`  - MultiPoint`
	`  coordinates:`
	`  type: array`
	`  minItems: 2`
	`  maxItems: 2`
	`  items:`
	`  title: Points along the route`
	`  type: array`
	`  minItems: 2`
	`  items:`
	`  type: number`
	`  preference:`
	`  type: string`
	`  default: fastest`
	`  enum:`
	`  - fastest`
	`  - shortest`

Permission 1	**/per/core/preference**
C	The `enum` and `default` values of `preference` in the schema MAY be extended to reflect the routing options supported by the server.

Note that additional members in the route definition can be ignored.

Requirement 4	**/req/core/conformance-values**

A	The content of the conformance declaration response at path `/conformance` SHALL list all values that the `preference` parameter supports, based upon the following OpenAPI 3.0 schema:
	`type: object`
	`properties:`
	`  http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/core:`
	`  type: object`
	`  required:`
	`  - values`
	`  properties:`
	`  values:`
	`  type: array`
	`  items:`
	`  minItems: 1`
	`  type: string`

Requirement 5	**/req/core/compute-route-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `201`.
B	The response SHALL include a header `Location` with the URI of the new route.

Requirement 6	**/req/core/error**
A	If the request does not conform to the requirements (e.g., the route definition is invalid) a response with status code `400` SHALL be returned.
A	If the request is valid, but the server is not able to process the request (e.g., the server has insufficient route network data for the request), a response with status code `422` SHALL be returned.

Example 4. Route definition

This requests the fastest route from Reagan Airport to the U.S. Captiol in Washington, D.C.

```
{
  "name": "Reagan Airport to Capitol",
  "waypoints": {
    "type": "MultiPoint",
    "coordinates": [
      [
        -77.037722,
        38.851444
      ],
      [
        -77.009003,
        38.889931
      ]
    ]
  },
  "preference": "fastest"
}
```

Example 5. New route response

The URI of the new route is `https://example.org/api/routing/v1/routes/hdg6g`.

```
HTTP/1.1 201 Created
Date: Fri, 26 Jul 2019 08:29:45 GMT
Location: https://example.org/api/routing/v1/routes/hdg6g
```

Permission 2	**/per/core/purge-routes**
A	Routing APIs may purge routes automatically.

Typically, routes will be removed after a reasonable time, for example, twelve hours after the route has last been accessed.

7.2.6. Route

Fetch a route

This operation returns the route with id `routeId`. The route content is described by the "Route Exchange Model (full)".

Requirement 7	/req/core/route-op
A	The server SHALL support the HTTP GET operation at the path `/routes/{routeId}` for each route referenced from the Routes resource at `/routes`.

Requirement 8	/req/core/route-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL conform to a requirements class of the Route Exchange Model.
C	By default (and this requirements class provides no mechanism to change the default), the content SHALL conform to the requirements class "Route Exchange Model (full)".
D	If elevation is provided for one coordinate in a route, all coordinates in the route SHALL include elevation information.
E	If the request included an `Accept-Language` header, the server SHALL try to honor the request and otherwise fall back to an available language.
F	The response SHALL include a `Content-Language` header with the language used for instructions and names, in particular road/street names.

A route is represented as a GeoJSON feature collection. Its contents will depend on the `status` of the route processing.

If the status is 'successful' the feature collection consists of the following information:

- A `name`, if one was provided with the route definition.
- A link to the canonical URI of the route and its definition (link relations `self` and `describedBy`)
- An array of features (the properties of each is to be decided)
  - The route overview feature. This has a LineString geometry of the complete route from start to end location.
  - The start point of the route with a Point geometry.
  - A feature for every segment of the route. This has a Point geometry representing the last point of the segment.
  - The end point of the route with a Point geometry.

If the status is 'accepted' (the request has been received, but processing has not yet started), 'running' (the routing is being computed) or 'failed' (there was an unspecified error computing the route) the feature collection has less information:

- The route overview has a `null` geometry.
- No segment features are included.

Example 6. A route

```
{
  "type": "FeatureCollection",
  "name": "Reagan Airport to Capitol",
  "status": "successful",
  "links": [
    {
      "href": "https://example.com/routes/hdg6g",
      "rel": "self",
      "type": "application/geo+json",
      "title": "this document"
    },
    {
      "href": "https://example.com/routes/hdg6g/definition",
      "rel": "describedBy",
      "type": "application/json",
      "title": "the route definition for this route"
    }
  ],
  "features": [
    {
      "type": "Feature",
      "id": 1,
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            -77.037722,
            38.851444
          ],
          ...,
          [
            -77.012520,
            38.889780
          ]
        ]
      },
      "properties": {
        "type": "route overview",
        "length_m": 8213,
        "duration_s": 483
      }
    },
    {
      "type": "Feature",
      "id": 2,
      "geometry": {
        "type": "Point",
        "coordinates": [
```

```
          -77.037722,
          38.851444
        ]
      },
      "properties": {
        "type": "start"
      }
    },
    {
      "type": "Feature",
      "id": 3,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -77.041674,
          38.871088
        ]
      },
      "properties": {
        "type": "segment",
        "length_m": 3314,
        "duration_s": 213,
        "instruction": "turn right",
        "roadName": "George Washington Memorial Pkwy",
        "maxHeight": 4.5,
        "speedLimit": 55,
        "speedLimitUnit": "mph"
      }
    },
    ...,
    {
      "type": "Feature",
      "id": 17,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -77.012520,
          38.889780
        ]
      },
      "properties": {
        "type": "segment",
        "length_m": 517,
        "duration_s": 73,
        "roadName": "First Street",
        "speedLimit": 35,
        "speedLimitUnit": "mph"
      }
    },
    {
      "type": "Feature",
```

```
      "id": 18,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -77.012520,
          38.889780
        ]
      },
      "properties": {
        "type": "end"
      }
    }
  ]
}
```

Example 7. A route that is still being computed


```
{
  "type": "FeatureCollection",
  "name": "Reagan Airport to Capitol",
  "status": "running",
  "links": [
    {
      "href": "https://example.com/routes/hdg6g",
      "rel": "self",
      "type": "application/geo+json",
      "title": "this document"
    },
    {
      "href": "https://example.com/routes/hdg6g/definition",
      "rel": "describedBy",
      "type": "application/json",
      "title": "the route definition for this route"
    }
  ],
  "features": [
    {
      "type": "Feature",
      "id": 1,
      "geometry": null,
      "properties": {
        "type": "route overview"
      }
    },
    {
      "type": "Feature",
      "id": 2,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -77.037722,
          38.851444
        ]
      },
      "properties": {
        "type": "start"
      }
    },
    {
      "type": "Feature",
      "id": 18,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -77.009003,
```

```
          38.889931
        ]
      },
      "properties": {
        "type": "end"
      }
    }
  ]
}
```


7.2.7. Route definition

Fetch the definition of a route

This operation returns the input parameters used to create the route with id `routeId`.

Requirement 9	**/req/core/route-definition-op**
A	The server SHALL support the HTTP GET operation at the path `/routes/{routeId}/definition` for each route referenced from the Routes resource at `/routes`.

Requirement 10	**/req/core/route-definition-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be identical to the content of the POST request to `/routes` when the route was created.

7.3. Requirements Class "Delete route"

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/delete-route	
Target type	Web API
Dependency	Requirements Class "Core"

7.3.1. Route

Delete a route

This operation deletes the route with identifier `routeId`. If the route is still in processing, the routing process is canceled.

Requirement 11	/req/delete-route/op
A	The server SHALL support the HTTP DELETE operation at the path `/routes/{routeId}` for each route referenced from the Routes resource at `/routes`.

Requirement 12	/req/delete-route/success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `204`.
B	The route SHALL be removed from the Routes resource (path `/routes`).
C	A GET request to `/routes/{routeId}` SHALL return a response with a HTTP status code `404`.

7.4. Requirements Class "Callback"

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/callback	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 13	/req/callback/input

A	The server SHALL support a member with the name "subscriber" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	`type: string`
	`format: uri`

B	The value of "subscriber" SHALL be a webhook (a HTTP(S) URI that accepts POST requests).

Requirement 14	/req/callback/success

A	If a webhook has been provided in a HTTP POST request to the path `/routes`, the server SHALL, after the computation of the route is finished, send a POST request to the webhook URI with the route according to the "Route Exchange Model (full)" as the content.

7.5. Requirements Class "Result set"

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/result-set	
Target type	Web API
Dependency	Requirements Class "Core"
Dependency	http://www.opengis.net/spec/rem/1.0/req/rem-segment-with-links

Requirement 15	**/req/result-set/input**

A	The server SHALL support a parameter with the name "resultSet" in GET requests to the path `/routes/{routeId}` with the following schema:
	`name: resultSet`
	`in: query`
	`schema:`
	`  type: string`
	`  enum:`
	`  - full`
	`  - overview`
	`  - segments`
	`  default: full`

Requirement 16	**/req/result-set/success**

A	If the `resultSet` parameter has been provided in the request, the server SHALL return the following after a successful execution of the request depending on the parameter value: • 'full' (default): the complete representation of the route according to requirements class "Route Exchange Model (full)". • 'overview': the route overview feature according to requirements class "Route Exchange Model (overview)". • 'segments': the first segment feature according to requirements class "Route Exchange Model (segment with links)"

If 'segments' is requested, the segment will include a link to the second segment (link relation `next`), if there is more than one segment. Every segment except the first and the last segment will include two links (link relations `prev` and `next`), except the last segment, which just has a `prev` link (unless there is only a single segment in which case there is no `prev` link).

It is up to the server how this is implemented and how segment URIs are minted. Options include another parameter to identify the segment by index or temporary, opaque URIs.

7.6. Requirements Class "Synchronous execution"

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/sync-mode	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 17	**/req/sync-mode/input**

A	The server SHALL support a parameter with the name "mode" in POST requests to the path `/routes` with the following schema:
	`name: mode`
	`in: query`
	`schema:`
	`  type: string`
	`  enum:`
	`  - async`
	`  - sync`
	`  default: async`

Requirement 18	/req/sync-mode/success
A	If the `mode` parameter has been provided in the request with the value 'sync', the server SHALL return the following after a successful computation of the requested route: • A response with the HTTP status code `200`. • The same content as to a GET request to path `/routes/{routeId}` once the route has the status 'successful'.
B	The computed route SHALL NOT be persistent on the server. No `routeId` will be assigned to the route.

If no parameter `mode` is provided or the parameter has the value 'async', the standard response is returned: a `201` response with the `Location` header pointing to the new Route resource.

If the route definition includes a member with name "subscriber" (see Requirements Class "Callback"), the information is ignored and no callback is executed.

7.7. Requirements Class "Intermediate waypoints"

Additional waypoints along the route between start and end to consider when computing the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/intermediate-waypoints	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 19	/req/intermediate-waypoints/input
A	The server SHALL support more than two points in the member with the name "waypoints" in the route definition in a HTTP POST request to the path `/routes` (i.e. `maxItems` may be removed from the schema definition or increased to a value larger than '2').

Requirement 20	/req/intermediate-waypoints/success
A	The computed route SHALL pass through all waypoints in the order in which they have been provided. "Pass through" means that the route overview line string geometry passes through the position or a position on the route network that is close to the waypoint.

7.8. Requirements Class "Height restriction"

A height restriction for vehicles in meters to consider when computing the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/max-height	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 21	**/req/max-height/input**

A	The server SHALL support a member with the name "maxHeight" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	`name: maxHeight`
	`in: query`
	`schema:`
	`  type: number`

Requirement 22	**/req/max-height/success**
A	The computed route SHALL be passable by vehicles with a height up to the value of "maxHeight" in meters.

7.9. Requirements Class "Load restriction"

A weight restriction for vehicles in tons to consider when computing the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/max-weight	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 23	**/req/max-weight/input**

A	The server SHALL support a member with the name "maxWeight" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	
	`name: maxWeight`
	`in: query`
	`schema:`
	`  type: number`

Requirement 24	**/req/max-weight/success**
A	The computed route SHALL be passable by vehicles with a weight up to the value of "maxWeight" in tons.

7.10. Requirements Class "Obstacles"

One or more polygons describing areas the route should avoid.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/obstacles	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 25	**/req/obstacles/input**

A	The server SHALL support a member with the name "obstacles" in the route definition in a HTTP POST request to the path `/routes` with the following schema (a GeoJSON MultiPolygon):
	
	`type: object`
	`required:`
	`  - type`
	`  - coordinates`
	`properties:`
	`  type:`
	`  type: string`
	`  enum:`
	`  - MultiPolygon`
	`  coordinates:`
	`  type: array`
	`  items:`
	`  type: array`
	`  items:`
	`  type: array`
	`  minItems: 4`
	`  items:`
	`  type: array`
	`  minItems: 2`
	`  items:`
	`  type: number`

Requirement 26	**/req/obstacles/success**
A	The computed route SHALL not pass through the polygons identified as obstacles.

NOTE This is a simple approach that is sufficient for the pilot. In general, the list of obstacles could also be a feature collection where every obstacle is a feature. Such a representation would be required, if the routing engine is able to handle obstacles with different characteristics/properties (for example, an obstacle is only valid for a certain time interval).

7.11. Requirements Class "Temporal constraints"

The time of departure or arrival. The default value is an immediate departure.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/time	
Target type	Web API

Dependency	Requirements Class "Core"

Requirement 27	**/req/time/input**

A	The server SHALL support a member with the name "when" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	
	`type: object`
	`required:`
	`  - timestamp`
	`properties:`
	`  timestamp:`
	`  type: string`
	`  format: date-time`
	`  example: "2019-05-23T19:06:32Z"`
	`  type:`
	`  type: string`
	`  default: departure`
	`  enum:`
	`  - departure`
	`  - arrival`

Requirement 28	**/req/time/success**
A	All temporal information in the route SHALL be based on the values in the "when" member (the time of departure or arrival, the default value is an immediate departure).

Recommendation 1	**/rec/time/success**
A	The route SHOULD consider the expected traffic situation at the time specified in the "when" member.

7.12. Requirements Class "Routing engine"

Select the routing engine to use for calculating the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/routing-engine	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 29	/req/routing-engine/input

A	The server SHALL support a member with the name "engine" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	
	`type: string`
	`enum:`
	`  - engineA`
	`  - engineB`
	`default: engineA`

B	The `enum` and `default` values in the schema SHALL be changed to reflect the routing engines supported by the server.

Requirement 30	/req/routing-engine/conformance-values

A	The content of the conformance declaration response at path `/conformance` SHALL list all values that the `engine` parameter supports, based upon the following OpenAPI 3.0 schema:
	
	`type: object`
	`properties:`
	`  http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-engine:`
	`    type: object`
	`    required:`
	`      - values`
	`    properties:`
	`      values:`
	`        type: array`
	`        items:`
	`          minItems: 1`
	`          type: string`

NOTE In the pilot, the engines are "Skymantics", "Ecere", and "HERE".

Requirement 31	/req/routing-engine/success
A	The route SHALL be computed with the selected routing engine.

7.13. Requirements Class "Routing algorithm"

Select the routing / graph solving algorithm to use for calculating the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/routing-algorithm	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 32	**/req/routing-algorithm/input**

A	The server SHALL support a member with the name "algorithm" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	`type: string`
	`enum:`
	`  - algorithmA`
	`  - algorithmB`
	`default: algorithmA`

B	The `enum` and `default` values in the schema SHALL be changed to reflect the algorithms supported by the server.

Requirement 33	**/req/routing-algorithm/conformance-values**

A	The content of the conformance declaration response at path `/conformance` SHALL list all values that the `algorithm` parameter supports, based upon the following OpenAPI 3.0 schema:
	`type: object`
	`properties:`
	`  http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/routing-algorithm:`
	`  type: object`
	`  required:`
	`  - values`
	`  properties:`
	`  values:`
	`  type: array`
	`  items:`
	`  minItems: 1`
	`  type: string`

Requirement 34	/req/routing-algorithm/success
A	The route SHALL be computed with the selected routing algorithm.

7.14. Requirements Class "Source dataset"

Select the source dataset for calculating the route.

Requirements Class	
http://www.opengis.net/spec/ogcapi-routes-1/1.0/req/source-dataset	
Target type	Web API
Dependency	Requirements Class "Core"

Requirement 35	/req/source-dataset/input

A	The server SHALL support a member with the name "dataset" in the route definition in a HTTP POST request to the path `/routes` with the following schema:
	`type: string`
	`enum:`
	`  - datasetA`
	`  - datasetB`
	`default: datasetA`

B	The `enum` and `default` values in the schema SHALL be changed to reflect the datasets supported by the server.

Requirement 36	/req/source-dataset/conformance-values

A	The content of the conformance declaration response at path `/conformance` SHALL list all values that the `dataset` parameter supports, based upon the following OpenAPI 3.0 schema:
	
	`type: object`
	`properties:`
	`  http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/source-dataset:`
	`  type: object`
	`  required:`
	`  - values`
	`  properties:`
	`  values:`
	`  type: array`
	`  items:`
	`  minItems: 1`
	`  type: string`

NOTE In the pilot, the datasets are "NSG", "OSM", and "HERE".

Requirement 37	**/req/source-dataset/success**
A	The route SHALL be computed with the selected dataset.

Chapter 8. Routing API as a profile of OGC API Processes

This clause specifies a second API option for a Routing API that supports more-or-less the same capabilities as the Routing API defined in the previous clause and re-uses the Route Exchange Model. However, it is based on the OGC API Processes standards and defined as a profile of that standard.

TODO: Do we want to keep the OGC API Processes option? If yes, we need to align with the latest draft. This is an old version based on an outdated draft of OGC API Processes. As discussed in the meeting on 2021-01-12, the preference is for a single API with explicit Route resources as described in Routing API . We will analyse, if the changes to OGC API Processes in recent month support that we can basically merge the two approaches without complicating the API. If that is not possible, an OGC API Processes profile for routing could be specified as a separate document, if there is interest, potentially by another group.

NOTE	This option is to a large extent specified by the OGC API - Processes candidate standard. It has additional requirements and conformance classes that mirror the ones from the first option. To avoid a lot of repetition, this clause currently does not define each of these classes formally like for the first option, but describes how each class of the first option is mapped to the OGC API Processes option.

8.1. Overview

The profile has the following characteristics:

- The resources are processes and jobs, consistent with the OGC API - Processes draft.
- The input to routing tasks (jobs) will be the start and end location plus optional routing parameters (intermediate waypoints, constraints, etc.), based on the route definition schema specified above.
- The output / result of routing tasks is using the Route Exchange Model in GeoJSON.

Note that one characteristic of this API option is that the input/output descriptions as part of the processOffering schema add an additional schema layer and describe another (WPS-specific) schema description language in addition to the JSON schema description in the OpenAPI definition. This also has the effect, that the schema of the routing parameters is not part of the API definition, but described as a response to a process resource in the API (`/processes/routing`).

The following table provides an overview of the resources. For details see the draft OGC API - Processes specification.

Note that the OGC API Common requirements apply to this option, too.

Table 2. Overview of resources and applicable HTTP methods in the WPS profile

Resource	Path	HTTP method	Document reference
Landing page	`/`	GET	Landing page
Conformance declaration	`/conformance`	GET	Conformance declaration
Processes	`/processes`	GET	Get processes
Process	`/processes/routing`	GET	Describe routing process
Jobs	`/processes/routing/jobs`	GET	Get routing jobs
		POST	Create new route
Job	`/processes/routing/jobs/{jobId}`	GET	Get routing job status
Result	`/processes/routing/jobs/{jobId}/result`	GET	Get route

8.2. Landing page

See API landing page, except that the "data" link references `/processes`.

8.3. Conformance declaration

The following is an example of the conformance declaration for the WPS option that implements all requirements classes. Note that the following classes of the Routing API option are not available in the WPS option as OGC API Processes does not support the required capabilities:

- Delete route
- Callback
- Result set
- Synchronous execution

Example 8. Conformance declaration


```
{
  "conformsTo": [
    "http://www.opengis.net/spec/WPS/2.0/req/service/binding/restjson/core",
    "http://www.opengis.net/spec/WPS/2.0/req/service/binding/restjson/oas30",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-intermediate-
waypoints",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-max-height",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-max-weight",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-obstacles",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-routing-engine",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-routing-algorithm",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-source-dataset",
    "http://www.opengis.net/spec/ogcapi-routes-1/1.0/conf/wps-time",
  ]
}
```


Note that the URIs of OGC API - Processes specification are outdated, but no newer URIs for conformance classes are available.

8.4. Get processes

A single process with id 'routing' is returned according to the processCollection schema.

Example 9. List of processes

```
{
  "processes": [ {
    "id": "routing",
    "title": "Compute routes",
    "description": "The process computes and creates a new route.

 At a minimum, a route is defined by two `waypoints`, the start and end
 point of the route.

 Every process supports at least 'fastest' and 'shortest' as the
 routing `preference`. The default value is 'fastest'.

An optional `name` for the route may be provided. The name will be
used as the title in links to the route and also included in the route
itself.

More parameters and routing constraints can optionally be provided
with the routing request:
* Source dataset to use when processing the route
* Routing engine to use when processing the route
* Routing algorithm to use when processing the route
* Obstacle requirements
* Height restriction
* Maximum load restriction
* Time of departure or arrival",
    "keywords": [ "routing" ],
    "version": "1.0",
    "jobControlOptions": [ "async-execute" ],
    "outputTransmission": [ "value" ],
    "links": [ {
      "href": "https://example.org/api/wps/v1/processes/routing",
      "rel": "???",
      "title": "execution endpoint"
    } ]
  } ]
}
```

8.5. Describe routing process

A description of the process with id 'routing' is returned according to the processOffering schema.

The `inputs` member will include all parameters that the WPS profile supports according to the declared conformance classes.

Example 10. Process description


```
{
  "inputs": [
    {
      "id": "waypoints",
      "title": "Waypoints",
      "description": "A list of points along the route. At least two points have
to be provided (start and end point).",
      "formats": [
        {
          "mimeType": "application/geo+json",
          "schema": "https://geojson.org/schema/MultiPoint.json"
        }
      ],
      "minOccurs": 1,
      "maxOccurs": 1
    },
    {
      "id": "preference",
      "title": "Routing preference",
      "description": "The routing preference.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "defaultValue": "fastest",
        "allowedValues": [
          "fastest",
          "shortest"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "maxHeight",
      "title": "Maximum height",
      "description": "A height restriction for vehicles in meters \nto consider
when computing the route.\n\nSupport for this parameter is not required and the
parameter may be\nremoved from the API definition.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
```

```
        "dataType": "double",
        "uom": {
          "name": "meter"
        }
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "maxWeight",
      "title": "Maximum weight",
      "description": "A weight restriction for vehicles in tons \nto consider when
computing the route.\n\nSupport for this parameter is not required and the
parameter may be\nremoved from the API definition.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "double",
        "uom": {
          "name": "tons"
        }
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "obstacle",
      "title": "???",
      "description": "???.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "defaultValue": "???",
        "allowedValues": [
          "???"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "dataset",
      "title": "source dataset",
      "description": "The source dataset to use for calculating the route.",
```

```
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "allowedValues": [
          "NSG",
          "OSM",
          "HERE"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "engine",
      "title": "routing engine",
      "description": "The routing engine to use for calculating the route.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "allowedValues": [
          "Skymantics",
          "Ecere",
          "HERE"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "algorithm",
      "title": "graph solving algorithm",
      "description": "The routing / graph solving algorithm to use for calculating
the route.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "defaultValue": "Dikjstra",
        "allowedValues": [
          "Dikjstra",
```

```
          "Floyd Marshall",
          "A*"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "when",
      "title": "time of departure or arrival",
      "description": "The time of departure or arrival. Default is \"now\".",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "dateTime"
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "deparr",
      "title": "departure",
      "description": "Specifies whether the value of `when` refers to the\ntime of
departure or arrival. Default is departure.",
      "formats": [
        {
          "mimeType": "text/plain"
        }
      ],
      "literalDataDomain": {
        "dataType": "string",
        "defaultValue": "departure",
        "allowedValues": [
          "departure",
          "arrival"
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    }
  ],
  "outputs": [
    {
      "id": "route",
      "title": "the route",
      "description": "The route is represented by a GeoJSON feature
collection\nthat contains the following information:\n\n* A `name`, if one was
provided with the route definition.\n* A link to the canonical URI of the route
```

```
and its definition\n(link relations `self` and `describedBy`)\n* An array of
features (the properties of each is to be decided)\n* The route overview feature.
This has a LineString \ngeometry of the complete route from start to end
location.\n* The start point of the route with a Point geometry.\n* A feature for
every segment of the route. This has a \nLineString geometry starting at the end
of the previous \nsegment (or, for the first segment, the start point).\n* The end
point of the route with a Point geometry.",
      "formats": [
        {
          "mimeType": "application/geo+json",
          "schema": "https://geojson.org/schema/FeatureCollection.json",
          "default": true
        }
      ]
    }
  ]
}
```


8.6. Get routing jobs

This operation just returns an object with a `jobs` member, which is an array of existing `jobId` values.

8.7. Create new route

This operation creates a new route. It is similar to the request in the Routing API option, except that the input/output descriptions according to OGC API Processes are used.

Example 11. New route

```
{
  "inputs": [
    {
      "id": "waypoints",
      "input": {
        "format": {
          "mimeType": "application/geo+json"
        },
        "value": {
          "inlineValue": {
            "type": "MultiPoint",
            "coordinates": [
              [
                36.1234515,
                32.6453783
              ],
              [
                36.1214698,
                32.655952
              ],
              [
                36.1247213,
                32.7106286
              ]
            ]
          }
        }
      }
    },
    {
      "id": "preference",
      "input": {
        "value": "fastest"
      }
    },
    {
      "id": "maxHeight",
      "input": {
        "value": "4.5",
        "uom": {
          "name": "meter"
        }
      }
    }
  ],
  "outputs": [
    {
      "id": "route",
```

```
      "output": {
        "format": {
          "mimeType": "application/geo+json"
        }
      },
      "transmissionMode": "value"
    }
  ]
}
```

8.8. Get routing job status

This operation informs about the status of the job with id `jobId`. It returns the status plus optionally a message and a progress estimate in percent.

The Routing API option currently does not support the message and the percent estimate.

8.9. Get route

The route according to the Route Exchange Model is returned, wrapped into objects and arrays according to OGC API Processes.

Example 12. A route

```
{
  "outputs": [
    {
      "id": "1",
      "value": {
        "inlineValue": { ... the route according to the Route Exchange Model ... }
      }
    }
  ]
}
```

Chapter 9. Media Types

TODO: List media types. For Route Exchange Model media types reference 21-001.

Chapter 10. Security considerations

The general Security Considerations for Web APIs are applicable to the Routing API, too. For example, see the Security Considerations section in OGC API - Features - Part 1: Core.

The following is a list of topics to be considered by implementers of the API:

- Fetching the list of routes (GET `/routes`) returns information about all routes on the server. In any operational, public deployment every user should probably only see their routes plus routes that others have shared with the user.
- The same applies to access to each route (GET `/route/{routeId}`) and their definition (GET `/route/{routeId}/definition`) as well as to the capability to cancel or delete a route (DELETE `/route/{routeId}`).

That is, any deployment that is not restricted to a closed group of users will typically require user accounts, access control to the API operations and a capability for sharing routes with the public or other users.

Annex A: Conformance Class Abstract Test Suite (Normative)

TODO

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2021-01-12	1.0.0-SNAPSHOT	C. Portele	all	initial version, based on Open Routing Pilot results

Annex C: Bibliography

TODO