WPS Routing API ER

Table of Contents

Publication Date: 2020-01-21

Approval Date: 2019-11-22

Submission Date: 2019-09-06

Reference number of this document: OGC 19-040

Reference URL for this document: http://www.opengis.net/doc/PER/WPSroutingAPI

Category: OGC Public Engineering Report

Editor: Christian Autermann

Title: WPS Routing API ER

OGC Public Engineering Report

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.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

The goal of this OGC WPS Routing API Engineering Report (ER) is to document the specification of an Application Programming Interface (API) which supports geographic routing. The specification includes two alternative approaches to such an API, one based on the current draft of the OGC API - Processes draft specification and another based on the OGC API principles (and the OGC API - Common draft specification). Both approaches facilitate a common Route Exchange Model.

Chapter 2. Executive Summary

This OGC WPS Routing API Engineering Report (ER) documents the requirements for interoperable web-based route computation and contains the specification of two alternative approaches to fulfill these requirements. One is based on the current draft of the OGC API - Processes draft specification while the other is based on the OGC API principles (and the OGC API - Common draft specification) and comprises a specialized API. Note that the OGC API - Processes draft specification is based on the existing Web Processing Service (WPS) standard. Both approaches facilitate a common Route Exchange Model that is based on GeoJSON [https://tools.ietf.org/html/rfc7946] and is specified in the Routing Pilot ER [1]. A visualization of this Route Exchange Model can be seen in Figure 1.

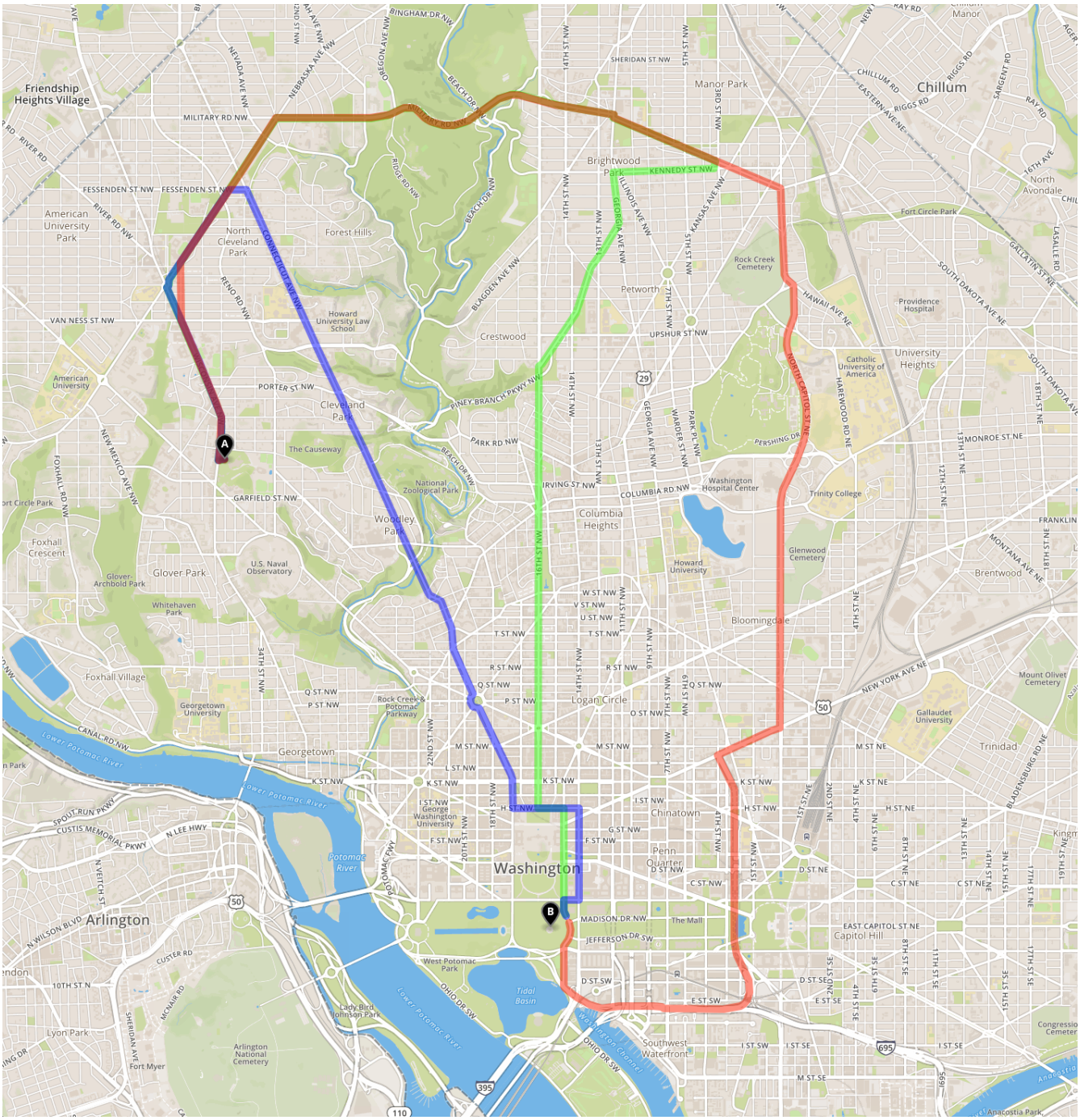

Figure 1. Routes from Washington National Cathedral to Washington Monument calculated with three different Routing APIs as specified in this ER, visualized using geojson.io [http://geojson.io/].

The approach using the OGC API - Processes specification has the advantage that routing APIs can

easily be integrated into process chains and generic clients. While this facilitates interoperability, it comes at a cost: the implementation of services and clients that are compliant to the OGC API - Processes specification comes with an overhead that may not be required in the process of route calculations. For example, support for bounding box or literal outputs are not used but required by the specification and the generation of user interfaces based on process descriptions is not a trivial task.

Contrarily, a specialized API based on the OGC API principles (and the OGC API - Common draft specification) can significantly reduce the burden of implementing new clients and APIs as the specification only requires what is really needed. While interoperability is maintained in terms of swapping one API against another, the integration of APIs into existing infrastructures would require facades or adapters.

If, on the other hand, existing service frameworks or clients that are compliant to the OGC API - Processes draft specification are in place, then the advantages of a specialized API diminish.

It was also discussed to adjust the OGC API - Processes draft specification to allow the creation of specialized, convenient APIs, such as the API specified in this ER, as profiles of the OGC API - Processes specification. This could be achieved by the using Hypermedia As The Engine Of Application State (HATEOAS) instead of enforcing specific endpoint paths.

Implementations of both approaches, findings, conclusions as well as recommendations can be found in the Routing Pilot Engineering Report [1].

2.1. Document contributor contact points

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

Contacts

Name	Organization	Role
Christian Autermann	52°North	Editor
Benjamin Proß	52°North	Contributor
Matthes Rieke	52°North	Contributor
Clemens Portele	interactive instruments GmbH	Contributor

2.2. Foreword

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.

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 14-065, OGC WPS 2.0 Interface Standard, version 2.0.2 (2018) [https://docs.opengeospatial.org/is/14-065/14-065.html]
- OGC: 17-069r3, OGC API - Features - Part 1: Core (2019) [http://docs.opengeospatial.org/is/17-069r3/17-069r3.html]
- OASIS: MQTT Version 5.0 (2019) [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html]

Chapter 4. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- **OGC API**

  the new OGC approach to web interfaces, currently represented by the new draft replacement to traditional WFS specification

- **OGC API WPS Profile**

  an OGC API web interface for a specific type of Web Processing Service

4.1. Abbreviated terms

- API Application Programming Interface
- DDIL Denied, Disrupted, Intermittent, and Limited
- HATEOAS Hypermedia As The Engine Of Application State
- HTTP Hypertext Transfer Protocol
- JSON JavaScript Object Notations
- REST Representational State Transfer
- WPS Web Processing Service
- YAML Yet Another Markup Language

Chapter 5. Overview

Chapter 6 presents the requirements for a web-based routing API as set by the sponsors of this pilot.

Chapter 7 gives a short overview of the OGC API - Processes draft specification, formerly known as WPS 2.0 REST/JSON Binding Extension, and in some previous literature referred to as a candidate WPS 3.0 standard.

Chapter 8 specifies the interface of two approaches to a web-based routing API, one based on the OGC API principles and the other based on the current draft of the OGC API for Processes.

Appendix A provides an example for the description of a process that conforms to the API defined in Chapter 8.

Appendix B lists example requests and responses of the API defined in Chapter 8.

Chapter 6. Pilot Requirements for Web-based Routing

This chapter will sum up the requirements for a web-based routing API as set by the sponsors of this pilot.

The bare minimum requirement for the Routing API is to asynchronously compute a route based on a start and end point. The resulting route has to be 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, the 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. OGC API - Processes

The current draft of OGC API - Processes specification (formerly known as "WPS 2.0 REST/JSON Binding Extension") specifies the interface to a general-purpose Web Processing Service (WPS) using a RESTful protocol transporting JSON encoded requests and responses. A WPS is a web service that enables the execution of computing processes and the retrieval of metadata describing their purpose and functionality. Typically these processes combine raster, vector, and/or coverage data with well-defined algorithms to produce new raster, vector, and/or coverage information. The document is an extension to the OGC WPS 2.0 Interface Standard [14-065]. It should be considered as another binding extension to HTTP/POST + XML and HTTP/GET + KVP as defined in Section 10 (Binding Extensions for WPS Operations) of the WPS 2.0 standard [2].

7.1. Added Value to Standardization Process

During the implementation of the APIs specified in Chapter 8 several issues in the current draft of the OGC API - Processes specification were found. These are described in detail in the Routing Pilot ER [1] and were reported to the WPS 2.0 SWG. By this, the pilot helped to mature the OGC API - Processes specification.

Chapter 8. Routing API

8.1. Overview

This clause specifies the Routing API as implemented in the OGC Open Routing API Pilot.

Two options are specified for the Routing API.

The first option is a Web API for routing based on the OGC API principles (and a future OGC API - Common standard) 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	Section 8.2.2
Conformance declaration	`/conformance`	GET	Section 8.2.3
Routes	`/routes`	GET	Section 8.2.5.1
		POST	Section 8.2.5.2
Route	`/routes/{routeId}`	GET	Section 8.2.6.1
		DELETE	Section 8.3.1.1
Route definition	`/routes/{routeId}/definition`	GET	Section 8.2.7.1

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.

The second API option is a Web API supporting more-or-less the same capabilities and re-using the Route Exchange Model, but based on the current draft of the OGC API - Processes specification.

This option is to a large extent specified by the OGC API - Processes draft specification. It has additional requirements and conformance classes that mirror the ones from the first option. To avoid a lot of repetition, this clause 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.

8.2. Requirements Class "Core"

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/core	
Target type	Web API
Dependency	OGC API - Common (Core)
Dependency	GeoJSON [https://tools.ietf.org/rfc/rfc7946.txt]
Dependency	http://www.opengis.net/orp/routing-api/1.0/req/rem-full

8.2.1. OGC API Common (Core)

At the time of writing, the OGC API - Common draft specification [3] 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 [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_root-op]
  - Requirement /req/core/root-success [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#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 [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_api-definition-op]
  - Permission /per/core/api-definition-uri [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#per_core_api-definition-uri]
  - Requirement /req/core/api-definition-success [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_api-definition-success]

    - Recommendation /rec/core/api-definition-oas [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_api-definition-oas]
- Conformance declaration
    - Requirement /req/core/conformance-op [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_conformance-op]
    - Requirement /req/core/conformance-success [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_conformance-success]
- Web API
    - Requirement /req/core/http [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_http]
    - Recommendation /rec/core/head [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_head]
    - Permission /per/core/additional-status-codes [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#per_core_additional-status-codes]
    - Requirement /req/core/query-param-unknown [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_query-param-unknown]
    - Requirement /req/core/query-param-invalid [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_query-param-invalid]
    - Recommendation /rec/core/etag [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_etag]
    - Recommendation /rec/core/cross-origin [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_cross-origin]
    - Recommendation /rec/core/link-header [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_link-header]
- Geometries
    - Requirement /req/core/crs84 [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#req_core_crs84]

The recommendation /rec/core/string-i18n [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#rec_core_string-i18n] is mainly implemented by the Content-Language header in the response to requests returning a route.

8.2.2. 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"
    }
  ]
}
```


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

8.2.4. 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.

8.2.5. 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/orapip/routing/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.

8.2.6. 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"
      }
    }
  ]
}
```


8.2.7. 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.

8.3. Requirements Class "Delete route"

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/delete-route	
Target type	Web API
Dependency	Section 8.2

8.3.1. 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`.

8.4. Requirements Class "Callback"

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/callback	
Target type	Web API

Dependency	Section 8.2

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.

8.5. Requirements Class "Result set"

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/result-set	
Target type	Web API
Dependency	Section 8.2
Dependency	http://www.opengis.net/orp/routing-api/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.

8.6. Requirements Class "Synchronous execution"

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/sync-mode	
Target type	Web API
Dependency	Section 8.2

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 Section 8.4), the information is ignored and no callback is executed.

8.7. 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/orp/routing-api/1.0/req/intermediate-waypoints	
Target type	Web API
Dependency	Section 8.2

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.

8.8. Requirements Class "Height restriction"

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

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/max-height	
Target type	Web API
Dependency	Section 8.2

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.

8.9. Requirements Class "Load restriction"

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

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/max-weight	
Target type	Web API
Dependency	Section 8.2

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.

8.10. Requirements Class "Obstacles"

One or more polygons describing areas the route should avoid.

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/obstacles	
Target type	Web API
Dependency	Section 8.2

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).

8.11. Requirements Class "Temporal constraints"

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

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/time	
Target type	Web API

Dependency	Section 8.2

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.

8.12. Requirements Class "Routing engine"

Select the routing engine to use for calculating the route.

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/routing-engine	
Target type	Web API
Dependency	Section 8.2

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/orapip/routing/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.

8.13. Requirements Class "Routing algorithm"

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

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/routing-algorithm	
Target type	Web API
Dependency	Section 8.2

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/orapip/routing/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.

8.14. Requirements Class "Source dataset"

Select the source dataset for calculating the route.

Requirements Class	
http://www.opengis.net/orp/routing-api/1.0/req/source-dataset	
Target type	Web API
Dependency	Section 8.2

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/orapip/routing/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.

8.15. WPS Profile of the Routing API

8.15.1. 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 [3] 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	Section 8.15.2
Conformance declaration	`/conformance`	GET	Section 8.15.3
Processes	`/processes`	GET	Section 8.15.4
Process	`/processes/routing`	GET	Section 8.15.5
Jobs	`/processes/routing/jobs`	GET	Section 8.15.6
		POST	Section 8.15.7
Job	`/processes/routing/jobs/{jobId}`	GET	Section 8.15.8
Result	`/processes/routing/jobs/{jobId}/result`	GET	Section 8.15.9

8.15.2. Landing page

See Section 8.2.2, except that the "data" link references `/processes`.

8.15.3. 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/orapip/routing/1.0/conf/wps-intermediate-waypoints",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-max-height",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-max-weight",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-obstacles",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-routing-engine",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-routing-algorithm",
    "http://www.opengis.net/orapip/routing/1.0/conf/wps-source-dataset",
    "http://www.opengis.net/orapip/routing/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.

8.15.4. Get processes

A single process with id 'routing' is returned according to the processCollection schema [https://raw.githubusercontent.com/opengeospatial/wps-rest-binding/master/core/openapi/schemas/processCollection.yaml].

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"
    } ]
  } ]
}
```

8.15.5. Describe routing process

A description of the process with id 'routing' is returned according to the processOffering schema [https://raw.githubusercontent.com/opengeospatial/wps-rest-binding/master/core/openapi/schemas/processOffering.yaml].

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
        }
      ]
    }
  ]
}
```

8.15.6. Get routing jobs

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

8.15.7. 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"
    }
  ]
}
```

8.15.8. 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.

8.15.9. 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 ... }
      }
    }
  ]
}
```

8.16. Discussion of the two options

The WPS profile option has several disadvantages from the view of a developer that is not familiar with the OGC API - Processes draft specification. Since developers that are not familiar with OGC standards and that do not have a comprehensive toolset supporting all kinds of OGC standards are an important target group of the OGC API family, this is relevant.

- "Process" and "Job" are not the resources you would expect in a dedicated routing API.
- The process description defines an OGC-specific schema language while OpenAPI already uses a schema language (JSON Schema). As a result, a developer has to understand the JSON Schema of the OGC-specific schema language first to understand the input and output schemas of the routing process.
- This also results in a more complex and verbose representation in the WPS profile option.

One option for migrating WPS capabilities to the OGC API family could be to mainly specify **guidance** and **patterns** for API building blocks for processing geospatial data. That is, building blocks for implementing asynchronous or synchronous processes, simple notifications using webhooks, more advanced notifications using MQTT or AsyncAPI, how to specify input and output schemas of processes, etc.

There are some aspects of the current WPS standard that might also lead to standards in the OGC API family. Known candidates are:

- One of the motivations for the WPS-specific process descriptions was to limit the schema complexity that WPS-implementations have to support. One approach could be to specify a simple JSON Schema profile for OGC API processing resources that would be based on the existing WPS standard. Since this would be an OGC standard, a key part of this would be spatial and temporal components. That is, existing WPS implementations would become backends enabling the rapid development of processing resources that follow the OGC API guidance.
- Another capability that needs more thought is WPS-T, i.e., the capability to deploy new processing resources. With the uptake of virtualization, containerization, cloud infrastructures, etc. it is obvious that this is more a general Information Technology (IT) topic than an OGC topic as there is nothing "geospatial" about deploying a new container on a server.

Finally, note that the current OGC API - Processes draft specification does not support some of the capabilities supported by the Routing API option:

- no support for retrieving the route definition;
- no support (yet) for canceling jobs, synchronous execution or webhooks.

Consider that the currently distinct elegant routing API (/routes) vs. traditional WPS (/processes/jobs) approaches could be harmonized by adjusting the OGC API - Processes draft specification, while always leaving the option to the service implementation not to expose the processing aspect (e.g. the /processes or equivalent resource), which could still be used in the background. This could apply equally well to other processing capabilities, such as server-side rendering (OGC API - Maps). OGC API - Processes adjustments which would enable this include:

- support for flexible resource paths being described in /processes (e.g. which could link to /routes)
- support for synchronous and/or asynchronous requests
- support for persistent, semi-persistent and/or temporary processing output (results) resources

More discussion about the two options and their implementations is documented in the Routing Pilot Engineering Report [1].

8.17. Security considerations

The general Security Considerations for Web APIs are applicable to the Routing API, too. For example, see the Security Considerations section [http://docs.opengeospatial.org/DRAFTS/17-069r2.html#_security_considerations] in OGC API - Features - Part 1: Core.

The pilot did not investigate aspects like security or privacy, the work instead focused on interoperability aspects. However, in any operational deployment security and privacy must be considered.

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.

Appendix A: Process Description


```
{
  "process": {
    "id": "routing",
    "title": "org.n52.routing",
    "version": "1.0.0",
    "jobControlOptions": ["async-execute", "sync-execute"],
    "outputTransmission": ["value", "reference"],
    "links": [
      {
        "href": "https://testbed.dev.52north.org/orp/processes/routing/jobs",
        "rel": "canonical",
        "title": "Execute endpoint"
      }
    ],
    "inputs": [
      {
        "id": "waypoints",
        "title": "The route waypoints",
        "description": "A list of points along the route. At least two points have to
be provided (start and end point).",
        "input": {
          "formats": [
            {
              "mimeType": "application/geo+json",
              "default": true
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "maxHeight",
        "title": "Maximum Height",
        "description": "The maximum height the vehicle has.",
        "input": {
          "literalDataDomains": [
            {
              "valueDefinition": {
                "anyValue": true
              },
              "dataType": {
                "name": "decimal",
                "reference": "https://www.w3.org/2001/XMLSchema-datatypes#decimal"
              }
            }
          ]
```

```
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "when",
      "title": "When",
      "description": "When to arrive or when to departure.",
      "input": {
        "formats": [
          {
            "default": true,
            "mimeType": "application/json"
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "maxWeight",
      "title": "Maximum Weight",
      "description": "The maximum weight the vehicle has.",
      "input": {
        "literalDataDomains": [
          {
            "valueDefinition": {
              "anyValue": true
            },
            "dataType": {
              "name": "decimal",
              "reference": "https://www.w3.org/2001/XMLSchema-datatypes#decimal"
            }
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "preference",
      "title": "Preference",
      "description": "The routing preference.",
      "input": {
        "literalDataDomains": [
          {
            "valueDefinition": ["fastest", "shortest"],
            "defaultValue": "fastest",
            "dataType": {
              "name": "string",
              "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
```

```
            }
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "obstacles",
      "title": "Obstacles",
      "description": "Obstacles along the route.",
      "input": {
        "formats": [
          {
            "default": true,
            "mimeType": "application/geo+json"
          },
          {
            "default": false,
            "mimeType": "application/geo+json"
          },
          {
            "default": false,
            "mimeType": "application/json"
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "name",
      "title": "The name",
      "description": "The name of the route.",
      "input": {
        "literalDataDomains": [
          {
            "valueDefinition": {
              "anyValue": true
            },
            "dataType": {
              "name": "string",
              "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
            }
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
```

```
        "id": "algorithm",
        "title": "Algorithm",
        "description": "The algorithm to use for route computation.",
        "input": {
          "literalDataDomains": [
            {
              "valueDefinition": [
                "astar",
                "astar-bi",
                "astar-ch",
                "dijkstra",
                "dijkstra-bi",
                "dijkstra-ch"
              ],
              "defaultValue": "astar-ch",
              "dataType": {
                "name": "string",
                "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
              }
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "dataset",
        "title": "Dataset",
        "description": "The dataset to use for route computation.",
        "input": {
          "literalDataDomains": [
            {
              "valueDefinition": ["HERE", "NSG", "OSM"],
              "defaultValue": "HERE",
              "dataType": {
                "name": "string",
                "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
              }
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      }
    ],
    "outputs": [
      {
        "id": "route",
        "title": "The Route.",
        "description": "The computed route.",
        "output": {
```

```
          "formats": [
            {
              "default": true,
              "mimeType": "application/geo+json"
            },
            {
              "default": false,
              "mimeType": "application/json"
            }
          ]
        }
      }
    ]
  }
}
```

Appendix B: Example Requests

B.1. GET Landing Page

B.1.1. Request

```
GET https://testbed.dev.52north.org/orp/here HTTP/1.1
Accept: application/json
```

B.1.2. Response


```
{
  "links": [
    {
      "href": "https://testbed.dev.52north.org/orp/here",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": "https://testbed.dev.52north.org/orp/here/api",
      "rel": "service",
      "type": "application/openapi+json;version=3.0",
      "title": "The API definition"
    },
    {
      "href": "https://testbed.dev.52north.org/orp/here/conformance",
      "rel": "conformance",
      "type": "application/json",
      "title": "WPS 2.0 REST/JSON Binding Extension conformance classes implemented by
this server"
    },
    {
      "href": "https://testbed.dev.52north.org/orp/here/processes",
      "rel": "processes",
      "type": "application/json",
      "title": "The processes offered by this server"
    },
    {
      "href": "https://testbed.dev.52north.org/orp/here/routes",
      "rel": "data",
      "type": "application/json",
      "title": "The routes."
    }
  ]
}
```

B.2. GET Conformance Classes

B.2.1. Request

```
GET https://testbed.dev.52north.org/orp/here/conformance HTTP/1.1
Accept: application/json
```

B.2.2. Response


```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-processes-1/1.0/conf/core",
    "http://www.opengis.net/orapip/routing/1.0/conf/core",
    "http://www.opengis.net/orapip/routing/1.0/conf/callback",
    "http://www.opengis.net/orapip/routing/1.0/conf/delete-route",
    "http://www.opengis.net/orapip/routing/1.0/conf/sync-mode",
    "http://www.opengis.net/orapip/routing/1.0/conf/max-height",
    "http://www.opengis.net/orapip/routing/1.0/conf/max-weight",
    "http://www.opengis.net/orapip/routing/1.0/conf/obstacles",
    "http://www.opengis.net/orapip/routing/1.0/conf/time",
    "http://www.opengis.net/orapip/routing/1.0/conf/intermediate-waypoints"
  ],
  "http://www.opengis.net/orapip/routing/1.0/conf/core": {
    "values": ["fastest", "shortest"]
  }
}
```


B.3. GET Processes

B.3.1. Request

```
GET https://testbed.dev.52north.org/orp/here/processes HTTP/1.1
Accept: application/json
```

B.3.2. Response

```
{
  "processes": [
    {
      "id": "routing",
      "title": "org.n52.routing.here",
      "version": "1.0.0",
      "jobControlOptions": ["async-execute", "sync-execute"],
      "outputTransmission": ["value", "reference"],
      "links": [
        {
          "href": "https://testbed.dev.52north.org/orp/here/processes/routing",
          "rel": "canonical",
          "type": "application/json",
          "title": "Process description"
        }
      ]
    }
  ]
}
```


B.4. GET Process Description

B.4.1. Request

```
GET https://testbed.dev.52north.org/orp/here/processes/routing HTTP/1.1
Accept: application/json
```

B.4.2. Response


```
{
  "process": {
    "id": "routing",
    "title": "org.n52.routing.here",
    "version": "1.0.0",
    "jobControlOptions": ["async-execute", "sync-execute"],
    "outputTransmission": ["value", "reference"],
    "links": [
      {
        "href": "https://testbed.dev.52north.org/orp/here/processes/routing/jobs",
        "rel": "canonical",
        "title": "Execute endpoint"
      }
    ],
    "inputs": [
      {
        "id": "waypoints",
```

```
        "title": "The route waypoints",
        "description": "A list of points along the route. At least two points have to
be provided (start and end point).",
        "input": {
          "formats": [
            {
              "mimeType": "application/geo+json",
              "default": true
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "maxHeight",
        "title": "Maximum Height",
        "description": "The maximum height the vehicle has.",
        "input": {
          "literalDataDomains": [
            {
              "valueDefinition": {
                "anyValue": true
              },
              "dataType": {
                "name": "decimal",
                "reference": "https://www.w3.org/2001/XMLSchema-datatypes#decimal"
              }
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "when",
        "title": "When",
        "description": "When to arrive or when to departure.",
        "input": {
          "formats": [
            {
              "default": true,
              "mimeType": "application/json"
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "maxWeight",
```

```
      "title": "Maximum Weight",
      "description": "The maximum weight the vehicle has.",
      "input": {
        "literalDataDomains": [
          {
            "valueDefinition": {
              "anyValue": true
            },
            "dataType": {
              "name": "decimal",
              "reference": "https://www.w3.org/2001/XMLSchema-datatypes#decimal"
            }
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "preference",
      "title": "Preference",
      "description": "The routing preference.",
      "input": {
        "literalDataDomains": [
          {
            "valueDefinition": ["fastest", "shortest"],
            "defaultValue": "fastest",
            "dataType": {
              "name": "string",
              "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
            }
          }
        ]
      },
      "minOccurs": 0,
      "maxOccurs": 1
    },
    {
      "id": "obstacles",
      "title": "Obstacles",
      "description": "Obstacles along the route.",
      "input": {
        "formats": [
          {
            "default": true,
            "mimeType": "application/geo+json"
          },
          {
            "default": false,
            "mimeType": "application/json"
          }
```

```
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      },
      {
        "id": "name",
        "title": "The name",
        "description": "The name of the route.",
        "input": {
          "literalDataDomains": [
            {
              "valueDefinition": {
                "anyValue": true
              },
              "dataType": {
                "name": "string",
                "reference": "https://www.w3.org/2001/XMLSchema-datatypes#string"
              }
            }
          ]
        },
        "minOccurs": 0,
        "maxOccurs": 1
      }
    ],
    "outputs": [
      {
        "id": "route",
        "title": "The Route.",
        "description": "The computed route.",
        "output": {
          "formats": [
            {
              "default": true,
              "mimeType": "application/geo+json"
            },
            {
              "default": false,
              "mimeType": "application/json"
            }
          ]
        }
      }
    ]
  }
}
```

B.5. POST Job

B.5.1. Request

```
POST https://testbed.dev.52north.org/orp/here/processes/routing/jobs HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
{
  "inputs": [
    {
      "id": "name",
      "input": { "value": "The name of the Route." }
    },
    {
      "id": "waypoints",
      "input": {
        "format": { "mimeType": "application/geo+json" },
        "value": {
          "inlineValue": {
            "type": "MultiPoint",
            "coordinates": [[-77.047712, 38.892346], [-76.99473, 38.902629]]
          }
        }
      }
    },
    {
      "id": "preference",
      "input": {
        "value": "fastest"
      }
    }
  ],
  "outputs": [
    {
      "id": "route",
      "format": { "mimeType": "application/geo+json" },
      "transmissionMode": "value"
    }
  ]
}
```

B.5.2. Response

```
HTTP/1.1 201
Content-Length: 0
Location: https://testbed.dev.52north.org/orp/here/processes/routing/jobs/68458445-
5d97-4d99-8a98-4ea17ac98842
```

B.6. GET Job Description

B.6.1. Request

```
GET https://testbed.dev.52north.org/orp/here/processes/routing/jobs/68458445-5d97-
4d99-8a98-4ea17ac98842 HTTP/1.1
Accept: application/json
```

B.6.2. Response

```
{
  "status" : "successful"
}
```

B.7. GET Job Result

B.7.1. Request

```
GET https://testbed.dev.52north.org/orp/here/processes/routing/jobs/68458445-5d97-
4d99-8a98-4ea17ac98842/result HTTP/1.1
content-type: application/json
```

B.7.2. Response

```
{
  "outputs": [
    {
      "id": "route",
      "value": {
        "inlineValue": {
          "type": "FeatureCollection",
          "status": "successful",
          "name": "The name of the Route.",
          "features": [
            {
              "type": "Feature",
              "properties": {
                "type": "overview",
```

```
            "length_m": 5588,
            "duration_s": 802
          },
          "geometry": {
            "type": "LineString",
            "bbox": [-77.0477092, 38.891741, -76.9947302, 38.9025235],
            "coordinates": [
              [-77.0477092, 38.8920951],
              [-77.0393193, 38.8920951],
              [-77.0388043, 38.8920844],
              [-77.0336545, 38.8920951],
              [-77.0260262, 38.8920844],
              [-77.0199108, 38.8920951],
              [-77.0192134, 38.8920844],
              [-77.0189667, 38.8921273],
              [-77.0186877, 38.8922453],
              [-77.0186126, 38.8923204],
              [-77.016499, 38.891741],
              [-77.0162416, 38.8918269],
              [-77.0161128, 38.8919234],
              [-77.0159733, 38.8919878],
              [-77.0157802, 38.8920414],
              [-77.0155871, 38.8920736],
              [-77.0153511, 38.8920951],
              [-77.0131946, 38.8920844],
              [-77.0128298, 38.892374],
              [-77.0118105, 38.8931251],
              [-77.009064, 38.8950992],
              [-77.009064, 38.8974702],
              [-77.009021, 38.8975561],
              [-77.0090103, 38.8976312],
              [-77.009021, 38.8985646],
              [-77.0089996, 38.8995731],
              [-77.0090103, 38.9002061],
              [-77.0083022, 38.9001954],
              [-77.0067036, 38.9002061],
              [-76.9949341, 38.9001954],
              [-76.9949448, 38.9025235],
              [-76.9947302, 38.9025235]
            ]
          }
        },
        {
          "type": "Feature",
          "properties": {
            "type": "start"
          },
          "geometry": {
            "type": "Point",
            "coordinates": [-77.047712, 38.892346]
          }
```

```
            },
            {
              "type": "Feature",
              "properties": {
                "type": "segment",
                "levelOfDetail": "visualization",
                "length_m": 1217,
                "duration_s": 155,
                "instructions": "continue",
                "text": "Head toward 21st St NW on Constitution Ave NW (US-50). Go for
1.2 km."
              },
              "geometry": {
                "type": "Point",
                "coordinates": [-77.0477092, 38.8920951]
              }
            },
            {
              "type": "Feature",
              "properties": {
                "type": "segment",
                "levelOfDetail": "visualization",
                "length_m": 1312,
                "duration_s": 151,
                "roadName": "Constitution Ave NW",
                "instructions": "continue",
                "text": "Continue on Constitution Ave NW (US-50). Go for 1.3 km."
              },
              "geometry": {
                "type": "Point",
                "coordinates": [-77.0336545, 38.8920951]
              }
            },
            {
              "type": "Feature",
              "properties": {
                "type": "segment",
                "levelOfDetail": "visualization",
                "length_m": 193,
                "duration_s": 39,
                "roadName": "Constitution Ave NW",
                "instructions": "right",
                "text": "Turn right onto Pennsylvania Ave NW. Go for 193 m."
              },
              "geometry": {
                "type": "Point",
                "coordinates": [-77.0186126, 38.8923204]
              }
            },
            {
              "type": "Feature",
```

```
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 297,
        "duration_s": 47,
        "roadName": "Pennsylvania Ave NW",
        "instructions": "left",
        "text": "Turn left onto Constitution Ave NW. Go for 297 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.016499, 38.891741]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 491,
        "duration_s": 75,
        "roadName": "Constitution Ave NW",
        "instructions": "left",
        "text": "Turn left onto Louisiana Ave NW. Go for 491 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0131946, 38.8920844]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 568,
        "duration_s": 141,
        "roadName": "Louisiana Ave NW",
        "instructions": "left",
        "text": "Turn left onto North Capitol St. Go for 568 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.009064, 38.8950992]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
```

```
        "length_m": 1221,
        "duration_s": 147,
        "roadName": "North Capitol St",
        "instructions": "right",
        "text": "Turn right onto H St NE. Go for 1.2 km."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0090103, 38.9002061]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 258,
        "duration_s": 41,
        "roadName": "H St NE",
        "instructions": "left",
        "text": "Turn left onto 8th St NE. Go for 258 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-76.9949341, 38.9001954]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 19,
        "duration_s": 6,
        "roadName": "8th St NE",
        "instructions": "right",
        "text": "Turn right onto K St NE. Go for 19 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-76.9949448, 38.9025235]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "end"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-76.99473, 38.902629]
```

```
            }
          }
        ]
      }
    }
  }
 ]
}
```

B.8. GET Routes

B.8.1. Request

```
GET https://testbed.dev.52north.org/orp/here/routes HTTP/1.1
Accept: application/json
```

B.8.2. Response

```
{
  "links": [
    {
      "href": "https://testbed.dev.52north.org/orp/here/routes",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href":
"https://testbed.dev.52north.org/orp/here/routes/5d64fe85adbe1d0001d74748",
      "rel": "item",
      "type": "application/geo+json",
      "title": "National Cathedral to Washington Monument"
    }
  ]
}
```

B.9. POST Route Definition

B.9.1. Request

```
GET https://testbed.dev.52north.org/orp/here/routes HTTP/1.1
Accept: application/json
```

```
{
  "name": "National Cathedral to Washington Monument",
  "preference": "fastest",
  "waypoints": {
    "type": "MultiPoint",
    "coordinates": [[-77.0721, 38.9309], [-77.0352, 38.8897]]
  }
}
```

B.9.2. Response

```
HTTP/1.1 201
Location: https://testbed.dev.52north.org/orp/here/routes/5d64fe85adbe1d0001d74748
```

B.10. GET Route

B.10.1. Request

```
GET https://testbed.dev.52north.org/orp/here/routes/5d64fe85adbe1d0001d74748 HTTP/1.1
Accept: application/geo+json
```

B.10.2. Response

```
{
  "type": "FeatureCollection",
  "status": "successful",
  "name": "National Cathedral to Washington Monument",
  "links": [
    {
      "href":
"https://testbed.dev.52north.org/orp/here/routes/5d64fe85adbe1d0001d74748",
      "rel": "self",
      "type": "application/geo+json"
    },
    {
      "href":
"https://testbed.dev.52north.org/orp/here/routes/5d64fe85adbe1d0001d74748/definition",
      "rel": "describedby",
      "type": "application/json"
    }
  ],
  "features": [
    {
      "type": "Feature",
      "properties": {
```

```
        "type": "overview",
        "length_m": 8294,
        "duration_s": 878
      },
      "geometry": {
        "type": "LineString",
        "bbox": [-77.0731902, 38.8867736, -77.0328605, 38.9309764],
        "coordinates": [
          [-77.0720636, 38.9309258],
          [-77.0721924, 38.9309764],
          [-77.0724285, 38.9309442],
          [-77.0725894, 38.9308798],
          [-77.0726538, 38.9308369],
          [-77.0728254, 38.9308584],
          [-77.0729864, 38.9297211],
          [-77.0731151, 38.9290237],
          [-77.0731902, 38.9280903],
          [-77.0731902, 38.9275324],
          [-77.0716774, 38.9275324],
          [-77.0685983, 38.9252687],
          [-77.0677829, 38.9246893],
          [-77.0675147, 38.9245391],
          [-77.066946, 38.9243674],
          [-77.0667315, 38.9243352],
          [-77.0663667, 38.9243245],
          [-77.0655835, 38.9241421],
          [-77.0652187, 38.9240134],
          [-77.0649827, 38.9239061],
          [-77.0646071, 38.9236701],
          [-77.064178, 38.9233375],
          [-77.0638454, 38.9230049],
          [-77.0637274, 38.9228547],
          [-77.0635772, 38.9225972],
          [-77.0634162, 38.9222646],
          [-77.0632231, 38.9216208],
          [-77.0631373, 38.9214063],
          [-77.0630729, 38.9213097],
          [-77.0629442, 38.9211702],
          [-77.062633, 38.9209127],
          [-77.0617425, 38.9202476],
          [-77.0594788, 38.9185953],
          [-77.0589745, 38.9182627],
          [-77.0582235, 38.9177048],
          [-77.0580733, 38.9175761],
          [-77.057333, 38.9170396],
          [-77.0576763, 38.9166534],
          [-77.0577836, 38.9164817],
          [-77.0578158, 38.9162135],
          [-77.0577836, 38.9160526],
          [-77.0577192, 38.9159131],
          [-77.0576227, 38.9158165],
```

```
          [-77.0572042, 38.9154947],
          [-77.0563245, 38.9148617],
          [-77.0559275, 38.9146042],
          [-77.0556164, 38.9143467],
          [-77.055155, 38.9141214],
          [-77.055037, 38.9140248],
          [-77.0549941, 38.9138961],
          [-77.0550048, 38.9136386],
          [-77.055037, 38.9134669],
          [-77.0550156, 38.9133811],
          [-77.0549726, 38.913306],
          [-77.0548868, 38.9132416],
          [-77.0545113, 38.9130485],
          [-77.0538998, 38.9126837],
          [-77.0533419, 38.9124799],
          [-77.0522904, 38.91204],
          [-77.0518398, 38.9116859],
          [-77.051636, 38.9115036],
          [-77.0514321, 38.911289],
          [-77.0510566, 38.9108491],
          [-77.050885, 38.9106774],
          [-77.0507777, 38.9106023],
          [-77.0505524, 38.910377],
          [-77.0504451, 38.9102054],
          [-77.0503914, 38.9099801],
          [-77.0503807, 38.9098299],
          [-77.0504129, 38.9096367],
          [-77.0504665, 38.9095294],
          [-77.0505846, 38.90939],
          [-77.0507133, 38.909272],
          [-77.0508206, 38.9091861],
          [-77.0509601, 38.9091218],
          [-77.0511854, 38.9090359],
          [-77.0513785, 38.9090145],
          [-77.0515716, 38.9090037],
          [-77.0525372, 38.9090359],
          [-77.0527518, 38.9090252],
          [-77.0530951, 38.9089501],
          [-77.0534492, 38.9088106],
          [-77.053889, 38.9085102],
          [-77.0542002, 38.9081776],
          [-77.0545113, 38.9077592],
          [-77.0552087, 38.9062035],
          [-77.0554876, 38.9056563],
          [-77.0557237, 38.9052486],
          [-77.0561743, 38.9046478],
          [-77.056421, 38.9042401],
          [-77.0565283, 38.9040148],
          [-77.056582, 38.9038646],
          [-77.0567, 38.9031136],
          [-77.0568824, 38.902663],
```

```
          [-77.0569575, 38.9023948],
          [-77.0571184, 38.9019978],
          [-77.0571828, 38.9017725],
          [-77.0572257, 38.9014292],
          [-77.057215, 38.9011288],
          [-77.0571721, 38.9009464],
          [-77.0570219, 38.9005709],
          [-77.0567107, 38.8996053],
          [-77.0564425, 38.8988543],
          [-77.0563781, 38.898586],
          [-77.0563459, 38.8983715],
          [-77.0563459, 38.8981247],
          [-77.0564103, 38.8976634],
          [-77.0564532, 38.8971269],
          [-77.0564747, 38.8951528],
          [-77.0564532, 38.8948309],
          [-77.0563996, 38.8945842],
          [-77.0563567, 38.8944447],
          [-77.0562387, 38.8941765],
          [-77.0557129, 38.8933396],
          [-77.0555627, 38.8931251],
          [-77.0554554, 38.8929105],
          [-77.055198, 38.8925242],
          [-77.0551014, 38.8923419],
          [-77.0549834, 38.8922453],
          [-77.0548654, 38.8922024],
          [-77.0547581, 38.8921702],
          [-77.0542109, 38.8920736],
          [-77.0540285, 38.8920093],
          [-77.0533526, 38.8916016],
          [-77.0530736, 38.8914621],
          [-77.052387, 38.8912153],
          [-77.0522261, 38.8911188],
          [-77.0521402, 38.8910329],
          [-77.0520759, 38.8909471],
          [-77.0520544, 38.890754],
          [-77.0520651, 38.8906467],
          [-77.0523334, 38.8901317],
          [-77.0525694, 38.8895845],
          [-77.052623, 38.8891768],
          [-77.0525908, 38.8889086],
          [-77.0525479, 38.8887584],
          [-77.0524728, 38.8885975],
          [-77.0522475, 38.8882434],
          [-77.0521188, 38.8880825],
          [-77.0519578, 38.8879108],
          [-77.0516467, 38.8876212],
          [-77.0513678, 38.8873851],
          [-77.0508742, 38.8872027],
          [-77.0505953, 38.8871276],
          [-77.0501983, 38.887074],
```

```
          [-77.0499408, 38.8870096],
          [-77.0491147, 38.886956],
          [-77.0480955, 38.8869345],
          [-77.0449626, 38.8868058],
          [-77.0429778, 38.8867736],
          [-77.0423341, 38.8868272],
          [-77.0417118, 38.8869131],
          [-77.0381391, 38.8869452],
          [-77.0367014, 38.8869452],
          [-77.0362508, 38.886956],
          [-77.0358753, 38.8869989],
          [-77.0352101, 38.8870847],
          [-77.0348454, 38.8871706],
          [-77.0346093, 38.8872564],
          [-77.0342767, 38.8874495],
          [-77.0338154, 38.8876748],
          [-77.0336866, 38.8877714],
          [-77.0336437, 38.8880289],
          [-77.0335472, 38.8881791],
          [-77.033118, 38.8886511],
          [-77.033, 38.888855],
          [-77.0329249, 38.8890588],
          [-77.0328605, 38.8895094],
          [-77.0329356, 38.8899279],
          [-77.0330107, 38.890121]
        ]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "start"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0721, 38.9309]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 72,
        "duration_s": 38,
        "instructions": "continue",
        "text": "Head northwest on North Rd. Go for 72 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0720636, 38.9309258]
```

```
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 372,
        "duration_s": 52,
        "roadName": "North Rd",
        "instructions": "left",
        "text": "Turn left onto Wisconsin Ave NW. Go for 372 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0728254, 38.9308584]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 131,
        "duration_s": 28,
        "roadName": "Wisconsin Ave NW",
        "instructions": "left",
        "text": "Turn left onto Garfield St NW. Go for 131 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0731902, 38.9275324]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 1754,
        "duration_s": 161,
        "roadName": "Garfield St NW",
        "instructions": "right",
        "text": "Turn right onto Massachusetts Ave NW. Go for 1.8 km."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0716774, 38.9275324]
      }
    },
    {
```

```
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 351,
        "duration_s": 49,
        "roadName": "Massachusetts Ave NW",
        "instructions": "right",
        "text": "Turn right onto Waterside Dr NW. Go for 351 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.057333, 38.9170396]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 3021,
        "duration_s": 269,
        "roadName": "Waterside Dr NW",
        "instructions": "continue",
        "text": "Take ramp onto Rock Creek and Potomac Pkwy NW. Go for 3.0 km."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0559275, 38.9146042]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 748,
        "duration_s": 89,
        "roadName": "Rock Creek and Potomac Pkwy NW",
        "text": "Keep left onto Ohio Dr SW. Go for 748 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0551014, 38.8923419]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
```

```
        "length_m": 1155,
        "duration_s": 104,
        "roadName": "Ohio Dr SW",
        "text": "Keep left onto Independence Ave SW. Go for 1.2 km."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0513678, 38.8873851]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 125,
        "duration_s": 15,
        "roadName": "Independence Ave SW",
        "text": "Keep right onto Independence Ave SW. Go for 125 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0381391, 38.8869452]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 283,
        "duration_s": 41,
        "roadName": "Independence Ave SW",
        "text": "Keep left onto Independence Ave SW. Go for 283 m."
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0367014, 38.8869452]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "segment",
        "levelOfDetail": "visualization",
        "length_m": 282,
        "duration_s": 32,
        "roadName": "Independence Ave SW",
        "text": "Turn slightly left onto 15th St SW. Go for 282 m."
      },
      "geometry": {
```

```
        "type": "Point",
        "coordinates": [-77.0336866, 38.8877714]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "type": "end"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [-77.0352, 38.8897]
      }
    }
  ]
}
```

Appendix C: Revision History

Table 3. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
May 28, 2019	B. Pross	0.1	all	initial version
July 08, 2019	B. Pross	0.2	overview	add OGC document number
August 5, 2019	M. Rieke	0.3	all	draft of document outline
August 6, 2019	C. Portele	0.4	routing api	include description of the routing API
September 2, 2019	C. Autermann	0.9	all	overview, summary and conclusions

Appendix D: Bibliography

1. Meek, S., Brown, T., Portele, C. eds: Routing Pilot ER. (2019).
2. Pross, B. ed: OGC WPS 2.0 REST/JSON Binding Extension. Open Geospatial Consortium, https://rawcdn.githack.com/opengeospatial/wps-rest-binding/master/docs/18-062.html (2018).
3. Heazel, C. ed: OGC API Common. Open Geospatial Consortium, https://rawcdn.githack.com/opengeospatial/oapi_common/master/19-xxx.html (2019).