Open Geospatial Consortium

Sprint Goals for OGC API - Moving Features -

The Vector Data Code Sprint

Taehoon Kim, AIST
12 July 2022

Sprint Goals for OGC API-MF

- Get opinions (and ideas) about draft documents and APIs.
  - **Draft documents and APIs**,
    - https://opengeospatial.github.io/ogcapi-movingfeatures/standard/standard_document.html
    - https://opengeospatial.github.io/ogcapi-movingfeatures/openapi/openapi-movingfeatures-1.html
  - **Validate** designed APIs with requirements,
    - *Missing contents;*
    - *Wrong contents;*
    - *Required contents;*
  - **Additional API** with **use-case**,
  - And so on…
- It will be very helpful for standardization work!

Sprint Goals for OGC API-MF (Continue)

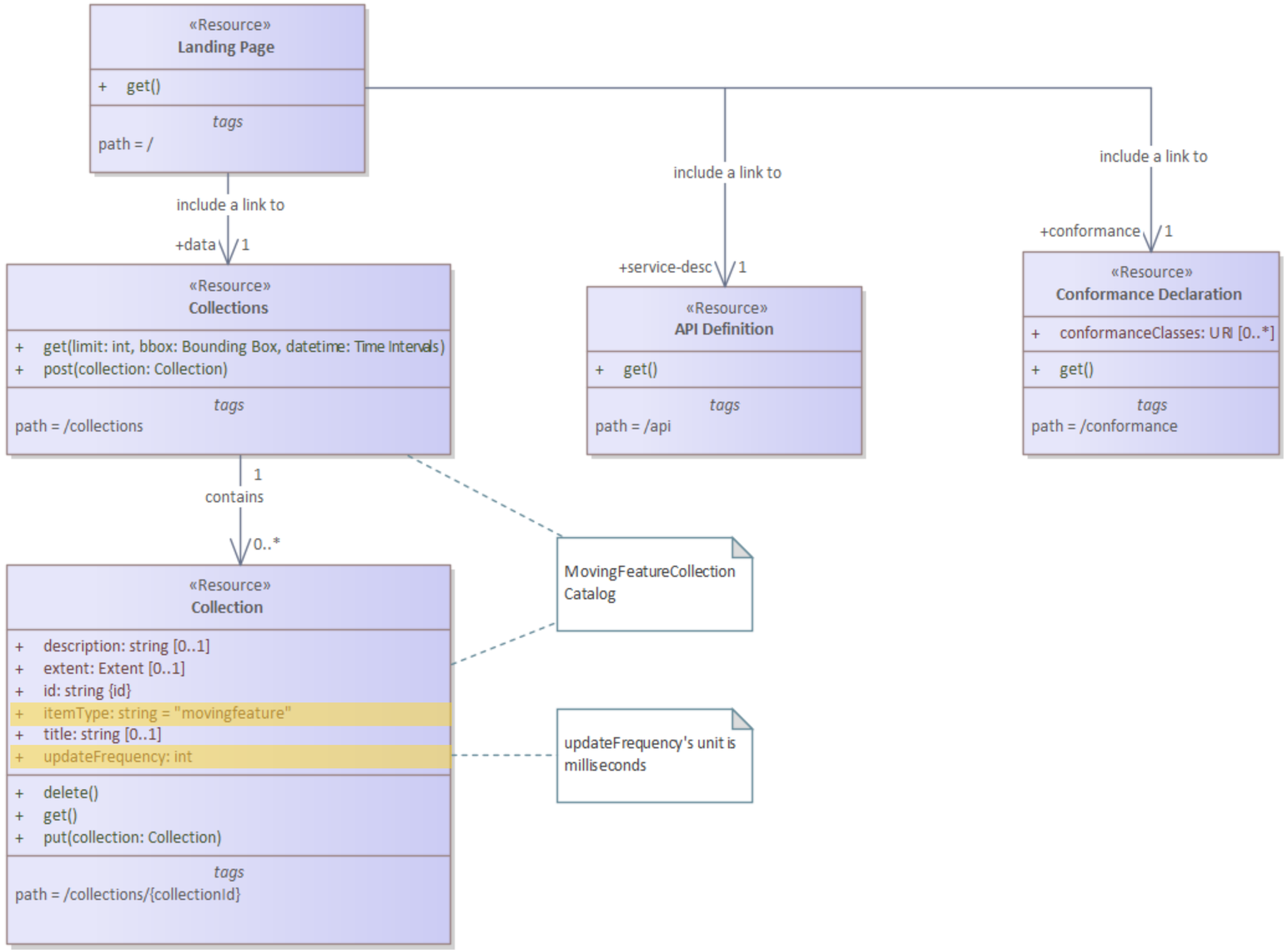
- New implementations for OGC API-MF with MF-JSON supported open-sources
  - MF-JSON: OGC Moving Features Encoding Extension–JSON (OGC 19-045r3)
  - Extend **GeoServer** (and **GeoTools**) to support MF-JSON and MF-APIs
- Annex: Current Open-Sources which support MF-JSON
  - Visualizations
    - **STINUUM** (https://github.com/aistairc/mf-cesium)
      - A space-time visual analysis tool of moving objects with Cesium
  - Spatio-temporal operations
    - **MobilityDB (**https://github.com/MobilityDB/MobilityDB)
      - An open-source geospatial trajectory data management & analysis platform
    - **MovingPandas (**https://github.com/anitagraser/movingpandas)
      - Implementation of Trajectory classes and functions built on top of GeoPandas

Overview of OGC API-*Moving Features*-Part 1

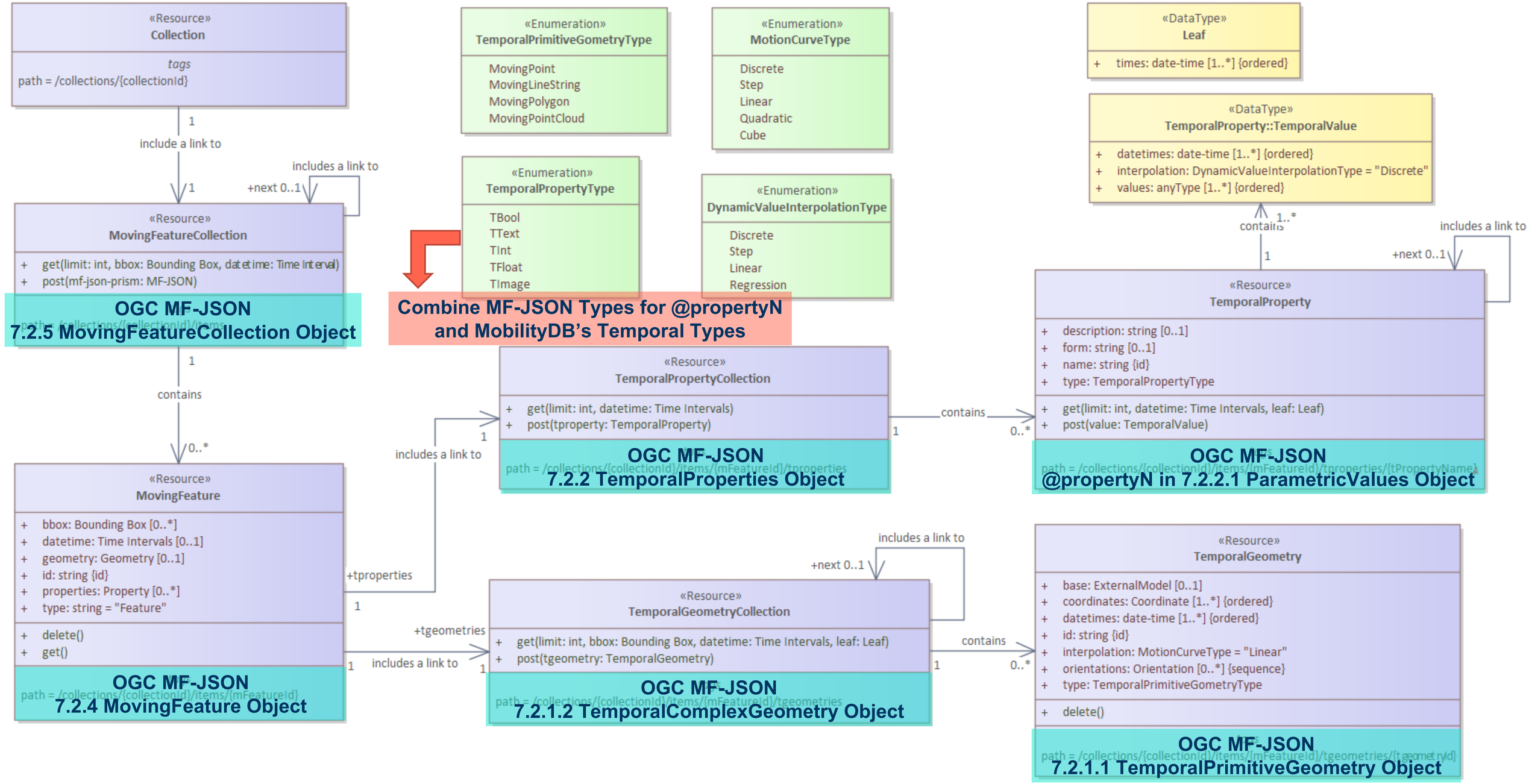
	Resource	Path	HTTP Method	Document Reference
Inherited from OGC API-Features	Landing page	/	GET	7.2 API Landing Page
	API definition	/api	GET	7.3 API Definition
	Conformance classes	/conformance	GET	7.4 Declaration of Conformance Classes
	Collections metadata	/collections	GET, POST	8.3 Resource Collections
	Collection instance metadata	/collections/{collectionId}	GET, DELETE, PUT	8.4 Resource Collection
Resources from MF-JSON	Moving features	/collections/{collectionId}/items	GET, POST	9.3 Resource MovingFeatures
	Moving feature instance	/collections/{collectionId}/items/{mFeatureId}	GET, DELETE	9.4 Resource MovingFeature
	Temporal geometry collection	/collections/{collectionId}/items/{mFeatureId}/tgeometries	GET, POST	9.5 Resource TemporalGeometryCollection
	Temporal geometry instance	/collections/{collectionId}/items/{mFeatureId}/tgeometries/{tGeometryId}	DELETE	9.6 Resource TemporalGeometry
	Temporal property collection	/collections/{collectionId}/items/{mFeatureId}/tproperties	GET, POST	9.7 Resource TemporalPropertyCollection
	Temporal property instance	/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyId}	GET, POST	9.8 Resource TemporalProperty

Inherited API-Features and Common

- There is **no modification** for supporting moving features.
- But there are two constraints:
  - The ***itemType*** of Collection should be "**movingfeature**"
  - The new property ***(updateFrequency***) is required to determine the continuous of temporal geometry
    - Update frequency: a time interval of sampling location

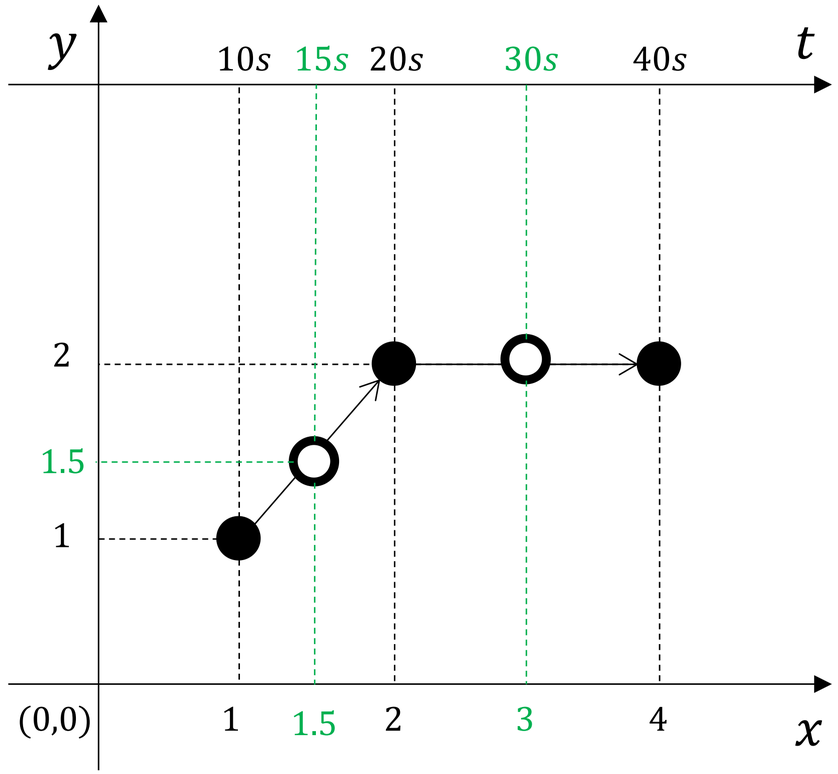


Conformance with OGC MF-JSON



Leaf

- Parameter 'Leaf' can be used in GET operation of **TemporalGeometryCollection** and **TemporalProperty**
  - ***pointAtTime*** operation in the [OGC Moving Feature Access standard]



<temporal geometry object>

```
{
 "id": "tgeom",
 "type": "MovingPoint",
 "datetimes": [
  "2021-09-14T12:00:10Z",
  "2021-09-14T12:00:20Z",
  "2021-09-14T12:00:40Z"
 ],
 "coordinates": [
  [1,1],
  [2,2],
  [4,2]
 ],
 "interpolation": "Linear",
}
```

If datetime= "2021-09-14T12%3A00%3A15Z%2F 2021-09-14T12%3A00%3A30Z", *then*

<TGeometries query response>

```
"temporalGeometries": [
 {
  "id": "tgeom",
  "type": "MovingPoint",
  "datetimes": [
   "2021-09-14T12:00:15Z",
   "2021-09-14T12:00:20Z",
   "2021-09-14T12:00:30Z"
  ],
  "coordinates": [
   [1.5,1.5],
   [2,2],
   [3,2]
  ],
  "interpolation": "Linear",
 }
],
...
```

If leaf=["2021-09-14T12:00:15Z", "2021-09-14T12:00:30Z"], *then*

<TGeometries query response>

```
"temporalGeometries": [
 {
  "id": "tgeom",
  "type": "MovingPoint",
  "datetimes": [
   "2021-09-14T12:00:15Z",
   "2021-09-14T12:00:30Z"
  ],
  "coordinates": [
   [1.5,1.5],
   [3,2]
  ],
  "interpolation": ”Discrete",
 }
],
...
```

More detailed information

- Please check our GitHub project
  - https://github.com/opengeospatial/ogcapi-movingfeatures
- If you have any questions or suggestions:
  - Make a new issue on our GitHub project!
    - https://github.com/opengeospatial/ogcapi-movingfeatures/issues
  - Feel free to ask me (kim.taehoon@aist.go.jp) via email or discord

Thank You

Community

500+ International Members

110+ Member Meetings

60+ Alliance and Liaison partners

50+ Standards Working Groups

45+ Domain Working Groups

25+ Years of Not for Profit Work

10+ Regional and Country Forums

Innovation

120+ Innovation Initiatives

380+ Technical reports

Quarterly Tech Trends monitoring

Standards

65+ Adopted Standards

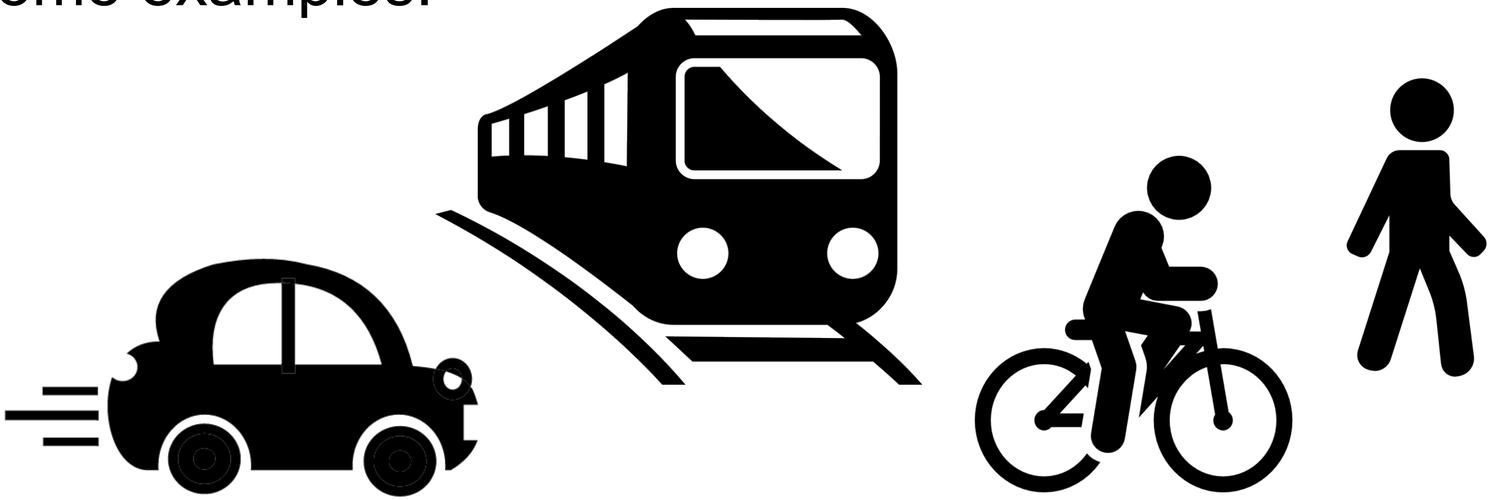
300+ products with 1000+ certified implementations

1,700,000+ Operational Data Sets

Using OGC Standards

Various types of moving features in the real world

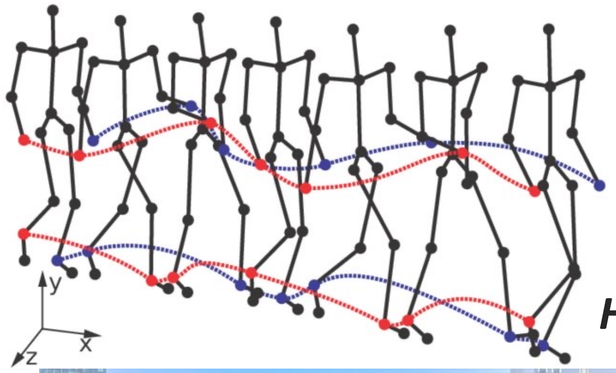
- **Moving Feature: feature whose location changes over time**
- We can easily imagine some examples:
  - Cars,
  - Trains,
  - Bicycles,
  - Pedestrians,
  - …



- Their positions changed over time, and they are usually represented by Points
  - How about its **shape**? With other representations, such as Curve, Surface, ...
  - How about its **properties**? Such as speeds, directions, capacities, …

Various representations with moving features

Curve type



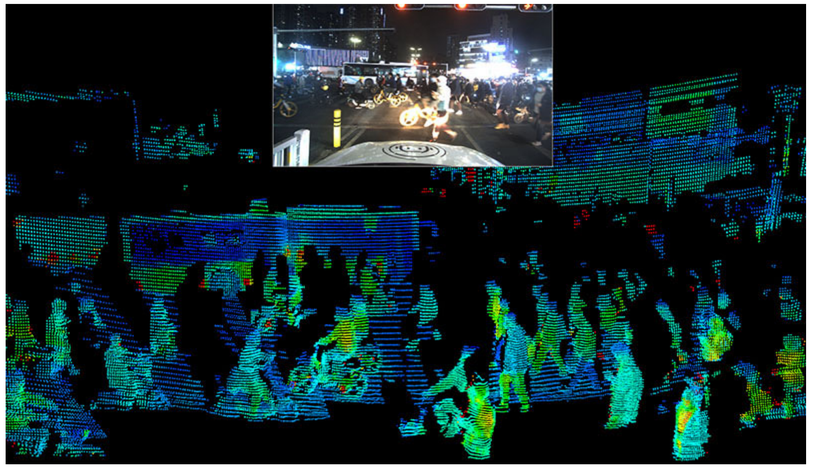
Human Motions[1]

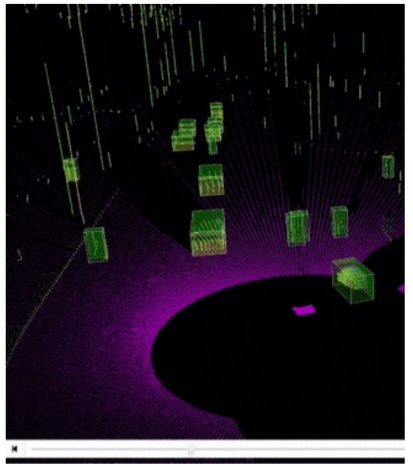
Trains[2]

Surface type

Typhoon[3]

Point Clouds





Pedestrians[4,5]

Image references:

1. Balazia, Michal, and Petr Sojka. "Learning robust features for gait recognition by maximum margin criterion." *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016.
2. https://matcha-jp.com/en/4409
3. https://github.com/aistairc/mf-cesium/wiki/Stinuum-Web-Manual
4. https://www.robosense.ai/en/tech-show-55
5. https://www.sama.com/3d-lidar-radar-point-cloud-annotation/

The property also can be changed over time

Video references: https://avs.auto/#/

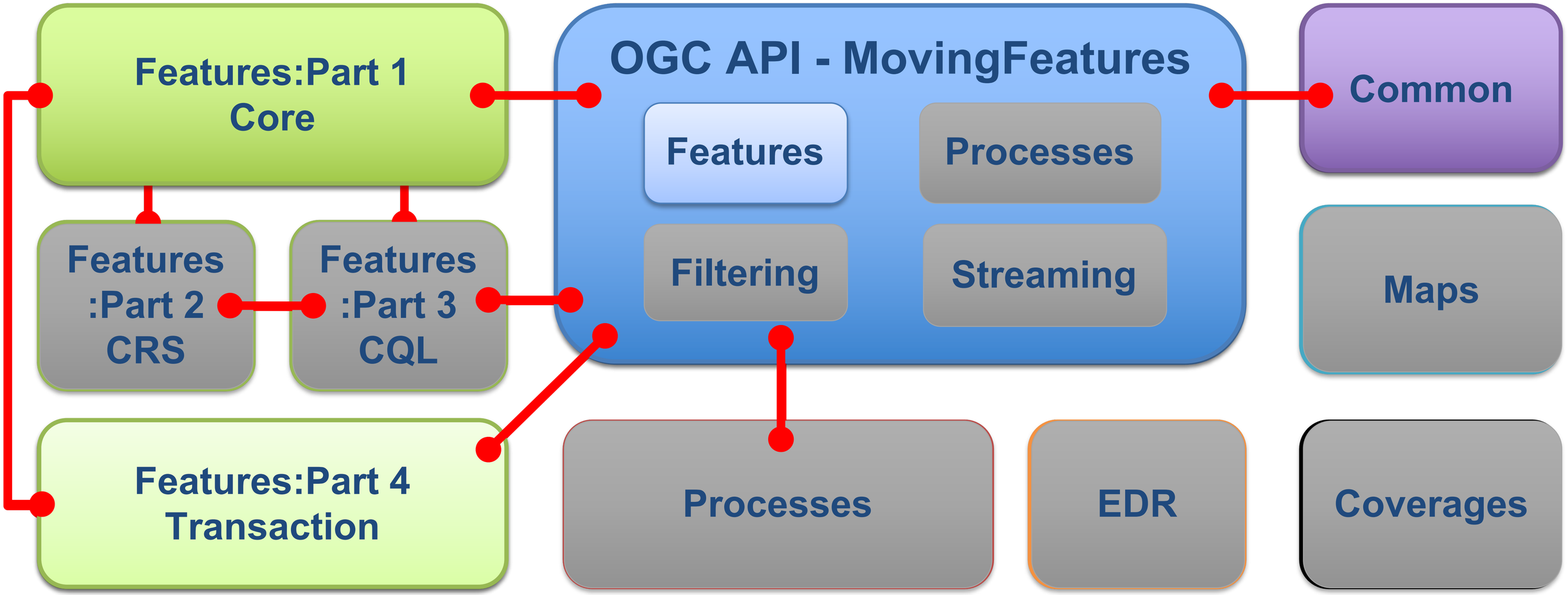
OGC Moving Features Encoding Extension–JSON

- https://docs.opengeospatial.org/is/19-045r3/19-045r3.html
- OGC Moving Features Encoding Extension–JSON (shortly, ***MF-JSON***) well covered these various cases
  - Core: MovingFeature (and MovingFeatureCollection) Object
  - Location: TemporalGeometry Object
    - Types: **MovingPoint**, **MovingLineString**, **MovingPolygon**, and **MovingPointCloud**
    - Interpolations: **Step**, **Linear**, **Quadratic**, **Cubic**
  - Properties: TemporalProperties Object
- There are other conceptual (and encoding) models for moving features, such as OGC MF XML encoding, CSV encoding, ISO 19141, and so on;
  - But they have not fully covered those cases.

OGC API–Moving Features

- **Part 1: Features Extension**
  - Moving feature is also a kind of feature.
    - Inherit the necessary APIs (and Resources) from OGC API-Features
  - Consider CRUD operations for MF-JSON objects
    - CRUD: Create, Read, Update, and Delete
- Part 2: Filtering Extension
  - based on OGC Moving Features Access Type A operations
- Part 3: Processes Extension
  - based on OGC Moving Features Access Type B&C operations
- Part 4: Streaming Extension

Relationship with other OGC APIs



OGC API–*Moving Features–Part 1:* Features extension

- **Overview:**
  - https://ogcapi.ogc.org/movingfeatures/
  - https://github.com/opengeospatial/ogcapi-movingfeatures
- **Draft document**:
  - https://opengeospatial.github.io/ogcapi-movingfeatures/standard/standard_document.html
- **OpenAPI**:
  - https://opengeospatial.github.io/ogcapi-movingfeatures/openapi/openapi-movingfeatures-1.html

- The goal of **OGC API – Moving Features Feature extension**:
  - Provide an interface for **Creating, Retrieving, Updating,** and **Deleting *Moving Features***, which conformance to the **OGC MF-JSON encoding standard**.