

Open Geospatial Consortium (OGC) Moving Features and Standards

International Standardization

Taehoon Kim (kim.taehoon@aist.go.jp)

Researcher, Main Editor of OGC API – Moving Features

Artificial Intelligence Research Center (AIRC)

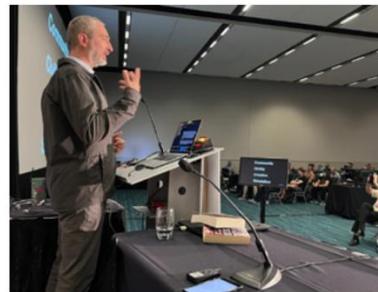
National Institute of Advanced Industrial Science and Technology (AIST)

The Open Geospatial Consortium (OGC)

community – standards - innovation

OGC is the one place for technology, innovation, standards, community, and more. We are a neutral playing field for innovation that eliminates friction across traditionally competitive organizations. Meet your future teammates, be a leader, or become an industry partner.

OGC's member-driven consensus process creates [royalty free, publicly available, open geospatial standards](#). Existing at the cutting edge, OGC actively analyzes and anticipates emerging trends, and runs an agile Research and Development (R&D) lab – the [OGC Innovation and Collaborative Solution and Innovation Program](#) – that builds and tests innovative prototype solutions to members' use cases.



For more than 28 years, Open Geospatial Consortium (OGC) has operated as a neutral forum where government, industry, nonprofits, and academia come together to engage in collective problem-solving around the critical issues of the day. As the global leader in location solutions and related data, OGC is the largest formal community of geospatial experts with a mission to make location information FAIR – Findable, Accessible, Interoperable, and Reusable – for an inclusive and sustainable future.

230+ members from industry

120+ government agencies

185+ universities & research orgs

70+ standards

100+ working groups



<https://www.ogc.org/about-ogc/ogc-member-list/>

OGC Membership



AIRBUS



EUROPEAN UNION
SATELLITE CENTRE

Analysis for decision making



AIRBUS

MAXAR



Hewlett Packard
Enterprise

HITACHI

SATELLLOGIC[®]

THALES

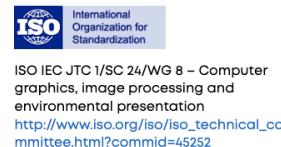
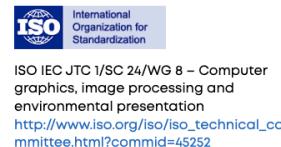
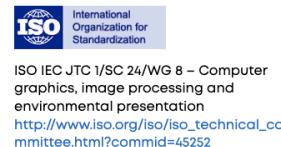
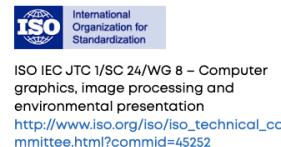
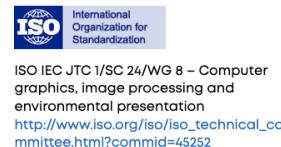
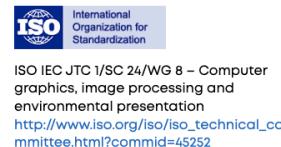
Courtesy of Nadine Alameh (CEO, OGC)

OGC Alliance Partners

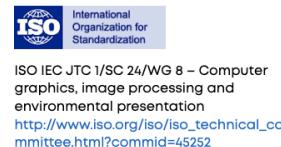
Cross-sectoral SDO collaboration activities



ISO Technical Committee 204 – Intelligent transport systems
http://www.iso.org/iso/standards_development/technical_committees/other_bodies/iso_technical_committee.htm?commid=54706



Source: <https://www.ogc.org/about-ogc/alliance-partners>



OGC Working Groups

• Domain Working Group (DWG)

- To provide a forum for discussion of key interoperability requirements and issues, discussion and review of implementation specifications, and presentations on key technology areas relevant to solving geospatial interoperability issues.
- Agriculture, Big Data, Earth Observation Exploitation Platform, etc.
- <https://www.ogc.org/about-ogc/committees/dwg/>

• Standard Working Group (SWG)

- To be working on a candidate standard prior to approval as an OGC standard or on making revisions to an existing OGC standard.
- Coverage, CRS, GeoTIFF, etc.
- <https://www.ogc.org/about-ogc/committees/swg/>

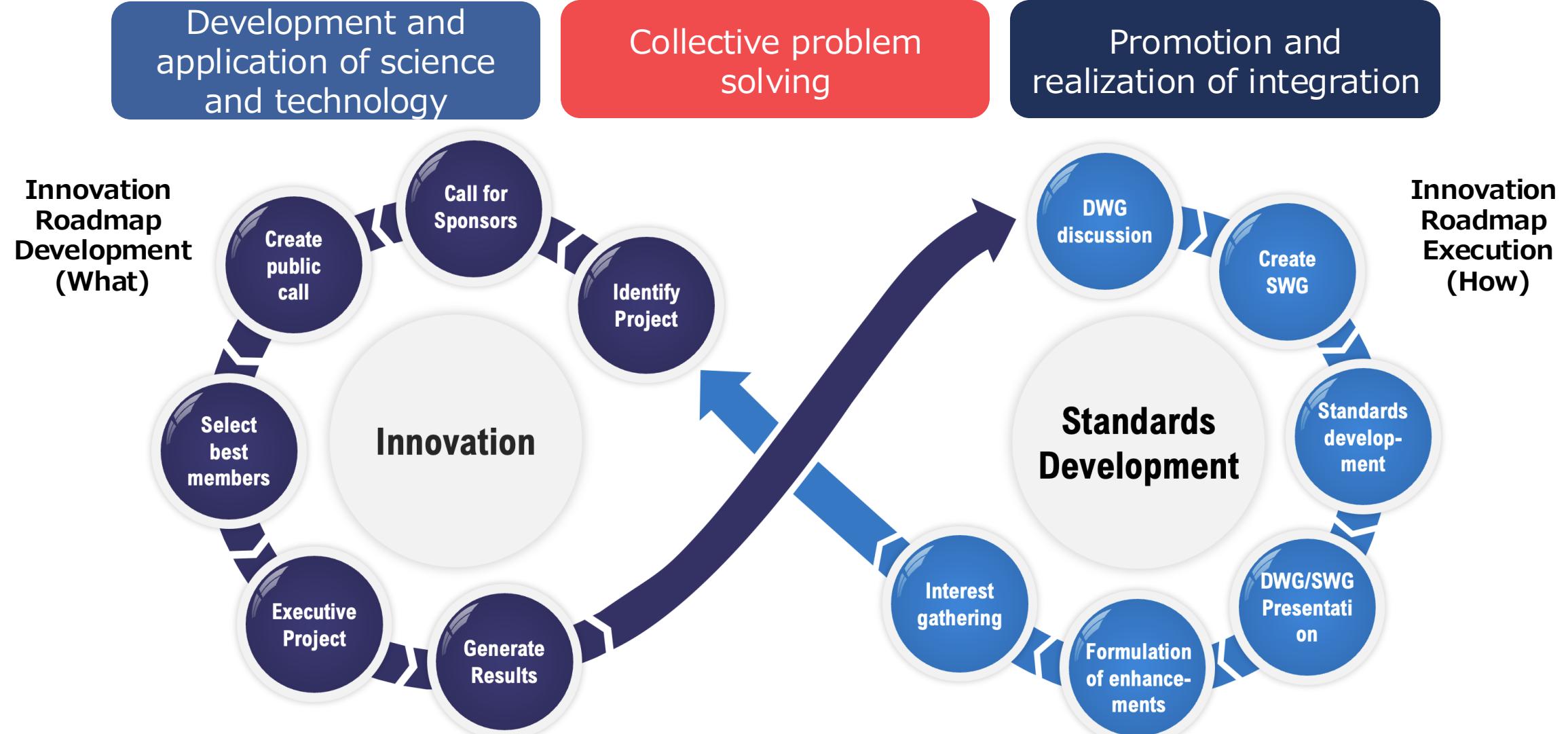
[3D Information Management DWG \(3DIM DWG\)](#)
[Agriculture DWG \(Agriculture DWG\)](#)
[Architecture DWG \(Arch DWG\)](#)
[Artificial Intelligence in Geoinformatics DWG \(GeoAI DWG\)](#)
[Aviation DWG \(Aviation DWG\)](#)
[Big Data DWG \(BigData DWG\)](#)
[Blockchain and Distributed Ledger Technologies DWG \(BDLT DWG\)](#)
[Citizen Science DWG \(Citizen Science\)](#)
[Coordinate Reference System DWG \(CRS DWG\)](#)
[Coverages DWG \(Coverages DWG\)](#)
[Data Preservation DWG \(PreservDWG\)](#)
[Data Quality DWG \(DQ DWG\)](#)
[Defense and Intelligence DWG \(D and I DWG\)](#)
[Discrete Global Grid Systems DWG \(DGGS DWG\)](#)
[Earth Observation Exploitation Platform DWG \(EO Ex Platform\)](#)
[Earth Systems Science DWG \(ESS DWG\)](#)
[Emergency and Disaster Management DWG \(EDM DWG\)](#)
[Energy and Utilities DWG \(EnergyUtilities\)](#)
[Geography Markup Language \(GML\) DWG \(GML DWG\)](#)
[Geoscience DWG \(Geoscience DWG\)](#)
[Geosemantics DWG \(Semantics\)](#)

[Health DWG \(Health DWG\)](#)
[Hydrology DWG \(Hydrology DWG\)](#)
[Interoperable Simulation and Gaming DWG \(ISG DWG\)](#)
[Land Administration DWG \(LandAdmin\)](#)
[Land and Infrastructure DWG \(LandInfraDWG\)](#)
[Marine DWG \(Marine DWG\)](#)
[Metadata and Catalog DWG \(MetaCat DWG\)](#)
[Meteorology & Oceanography DWG \(Met Ocean DWG\)](#)
[Mobile Location Services DWG \(MLSDWG\)](#)
[Perspective Imagery DWG \(PerspecImagervD\)](#)
[Point Cloud DWG \(Point Cloud DWG\)](#)
[Portrayal DWG \(Portrayal DWG\)](#)
[Quality of Service and Experience DWG \(QoSE DWG\)](#)
[Security DWG \(SecurityDWG\)](#)
[Sensor Web Enablement DWG \(SensorWeb DWG\)](#)
[Smart Cities DWG \(SmartCities DWG\)](#)
[Statistical DWG \(Statistical DWG\)](#)
[Temporal DWG \(Temporal DWG\)](#)
[Uninhabited Systems \(UxS\) DWG \(UxS DWG\)](#)
[University DWG \(Univ DWG\)](#)
[Workflow DWG \(Workflow DWG\)](#)

[3D GeoVolumes SWG \(3DGeoVol SWG\)](#)
[3D Portrayal SWG \(3DP SWG\)](#)
[CDB SWG \(CDB SWG\)](#)
[CityGML SWG \(CityGML SWG\)](#)
[Coverages SWG \(CoveragesSWG\)](#)
[CRS SWG \(CRS SWG\)](#)
[CRS Well Known Text SWG \(CRS WKT SWG\)](#)
[Discrete Global Grid Systems SWG \(DGGS SWG\)](#)
[Environmental Data Retrieval API SWG \(EDR-API SWG\)](#)
[EO Product Metadata and OpenSearch SWG \(EO PMOS SWG\)](#)
[Features and Geometries JSON SWG \(FeatGeoJSON SWG\)](#)
[Features API SWG \(FeatAPI SWG\)](#)
[GeoAPI SWG \(GeoAPI SWG\)](#)
[Geocoding API SWG \(GeocodeAPISWG\)](#)
[GeoPackage SWG \(GeoPackage SWG\)](#)
[GeoPose SWG \(GeoPose SWG\)](#)
[GeoSciML SWG \(GeoSciML SWG\)](#)
[GeoSPARQL SWG \(GeoSPARQL SWG\)](#)
[Geospatial User Feedback SWG \(GUFSwga\)](#)
[GeoSyncrhonization 1.0 SWG \(Geosync SWG\)](#)
[GeoTIFF SWG \(GeoTIFF SWG\)](#)
[GeoXACML SWG \(GeoXACML SWG\)](#)
[GML 3.3 SWG \(GML 3.3 SWG\)](#)
[GMLJP2 SWG \(GMLJP2-SWG\)](#)
[Groundwater SWG \(GroundwaterSWG\)](#)
[HDF SWG \(HDF SWG\)](#)
[Hydrologic Features SWG \(HydroFeat SWG\)](#)

[IndoorGML SWG \(IndoorGML SWG\)](#)
[KML 2.3 SWG \(KML SWG\)](#)
[Land and Infrastructure SWG \(LandInfraSWG\)](#)
[Moving Features SWG \(MovFeat SWG\)](#)
[MUDDI SWG \(MUDDI SWG\)](#)
[NetCDF SWG \(NetCDFSWG\)](#)
[O&M SWG \(OM SWG\)](#)
[OGC API - Common SWG \(OGC API-Common\)](#)
[OGC API - Maps SWG \(OGC API - Maps\)](#)
[OGC API - Processes SWG \(OAPIProc SWG\)](#)
[OGC API - Records SWG \(API Records SWG\)](#)
[OGC API - Styles SWG \(Styles API SWG\)](#)
[OGC API - Tiles SWG \(OAPITileSWG\)](#)
[OWS Common - Security SWG \(ComSecuritySWG\)](#)
[OWS Context SWG \(OWScontextSWG\)](#)
[PipelineML SWG \(PipeML SWG\)](#)
[Points of Interest SWG \(Pol SWG\)](#)
[PubSub SWG \(PubSub SWG\)](#)
[Routing SWG \(Routing SWG\)](#)
[Sensor Model Language \(SensorML\) 2.0 SWG \(SensorML2.0SWG\)](#)
[SensorThings SWG \(SensorThings\)](#)
[Simple Features SWG \(SF SWG\)](#)
[Styles and Symbology Encoding SWG \(Styles SE SWG\)](#)
[Temporal WKT for Calendars SWG \(TemporalWKT\)](#)
[TimeSeriesML SWG \(TimeSeriesML\)](#)
[Training Data Markup Language for AI SWG \(TrainingDML SWG\)](#)
[WaterML 2.0 SWG \(WaterML2.0SWG\)](#)

OGC Innovation Continuum



[SOURCE] https://portal.ogc.org/files/?artifact_id=92756

OGC Innovation Projects

We solve problems - together



100+

Completed Projects

25+

Active Initiatives

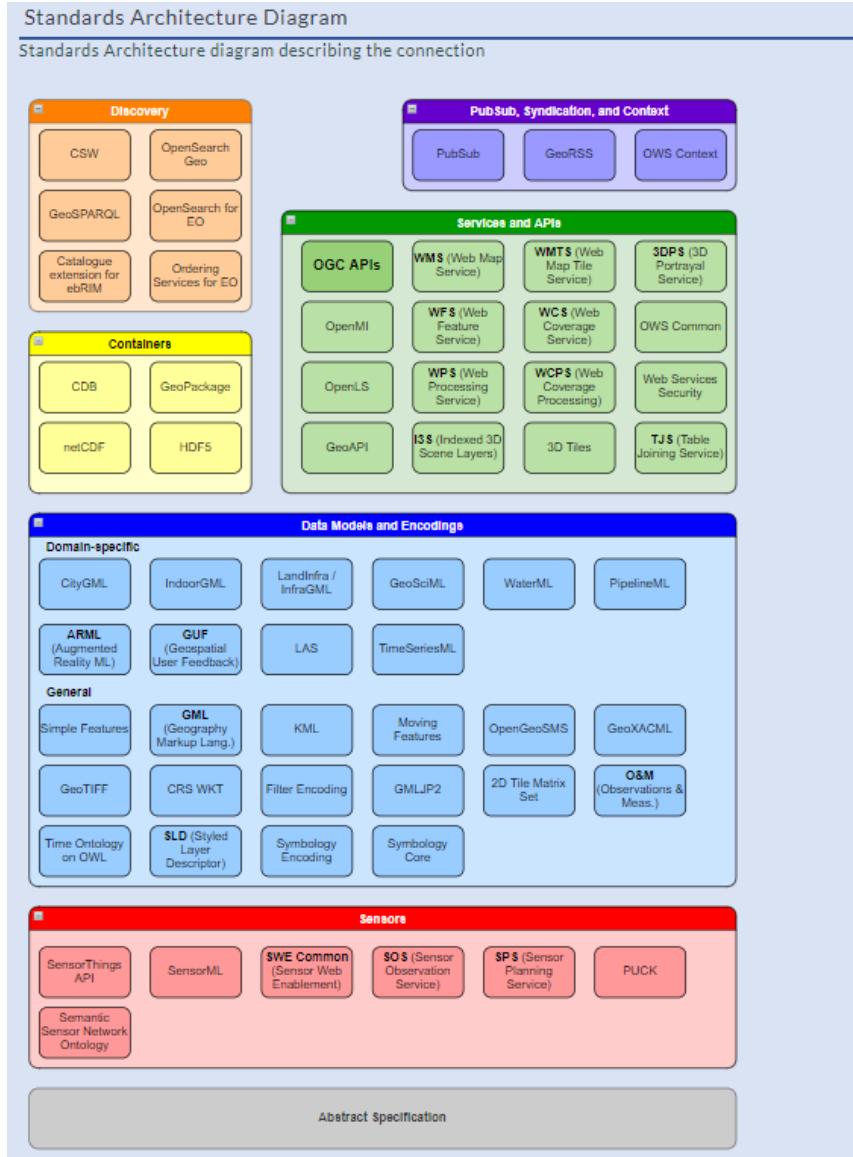
1.8M

Redistributed to OGC members

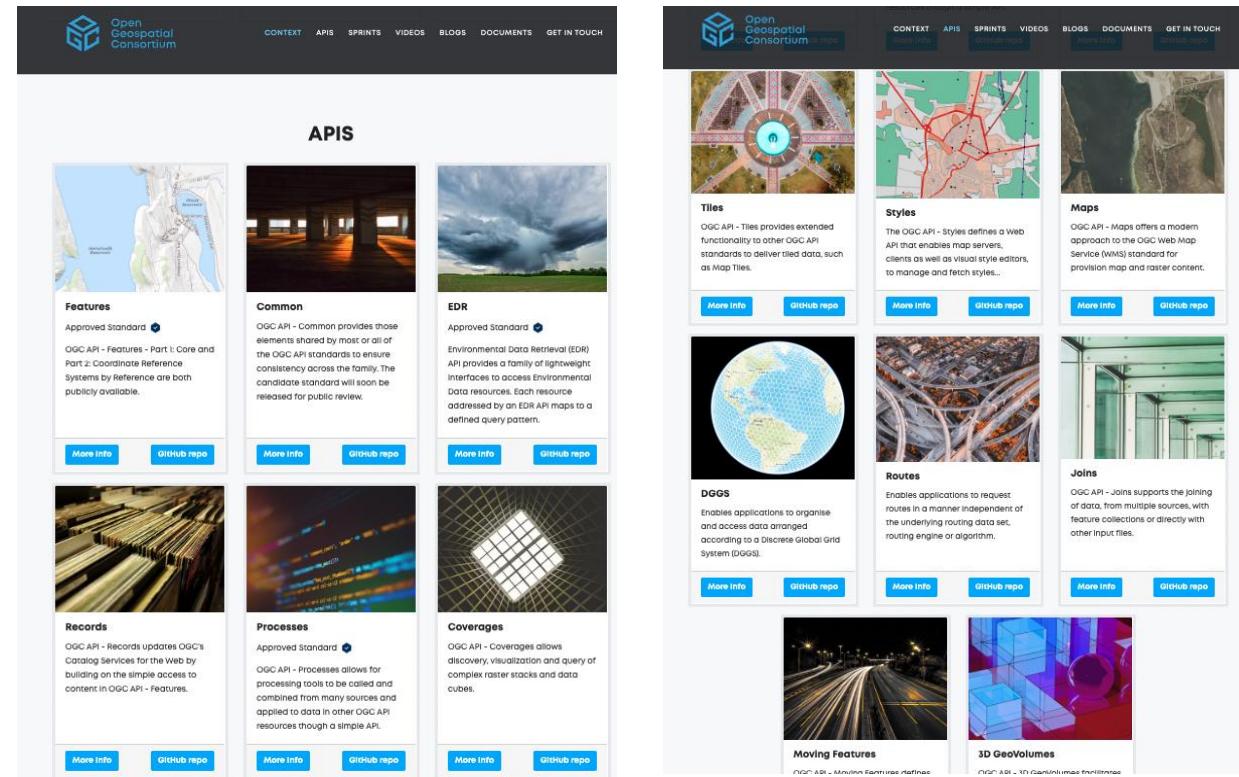
OGC Brings together the experts and the resources for collective problem solving

Courtesy of Nadine Alameh (CEO, OGC)

Digital Twins – standards to connect silos



<https://ogcapi.ogc.org/>



The screenshot shows the OGC APIs website interface, featuring the Open Geospatial Consortium logo at the top. The main content area is titled "APIS" and displays a grid of service categories with corresponding images and descriptions. The categories include:

- Features:** Approved Standard. Description: OGC API - Features - Part 1: Core and Part 2: Coordinate Reference Systems by Reference are both publicly available.
- Common:** Approved Standard. Description: OGC API - Common provides those elements shared by most or all of the OGC API standards to ensure consistency across the family. The candidate standard will soon be released for public review.
- EDR:** Approved Standard. Description: Environmental Data Retrieval (EDR) API provides a family of lightweight interfaces to access Environmental Data resources. Each resource addressed by an EDR API maps to a defined query pattern.
- Records:** Description: OGC API - Records updates OGC's Catalog Services for the Web by building on the simple access to content in OGC API - Features.
- Processes:** Approved Standard. Description: OGC API - Processes allows for processing tools to be called and combined from many sources and applied to data in other OGC API resources through a simple API.
- Coverages:** Description: OGC API - Coverages allows discovery, visualization and query of complex raster stacks and data cubes.
- Tiles:** Description: OGC API - Tiles provides extended functionality to other OGC API standards to deliver tiled data, such as Map Tiles.
- Styles:** Description: The OGC API - Styles defines a web API that enables map servers, clients as well as visual style editors, to manage and fetch styles...
- Maps:** Description: OGC API - Maps offers a modern approach to the OGC Web Map Service (WMS) standard for provision map and raster content.
- Routes:** Description: Enables applications to request and access data arranged according to a Discrete Global Grid System (DGGS).
- Joins:** Description: OGC API - Joins supports the joining of data, from multiple sources, with feature collections or directly with other input files.
- Moving Features:** Description: OGC API - Moving Features defines...
- 3D GeoVolumes:** Description: OGC API - 3D GeoVolumes...

Courtesy of Nadine Alameh (OGC)

Standards



Standards

Best practices

Readiness guides

Developer resources

Definition Server

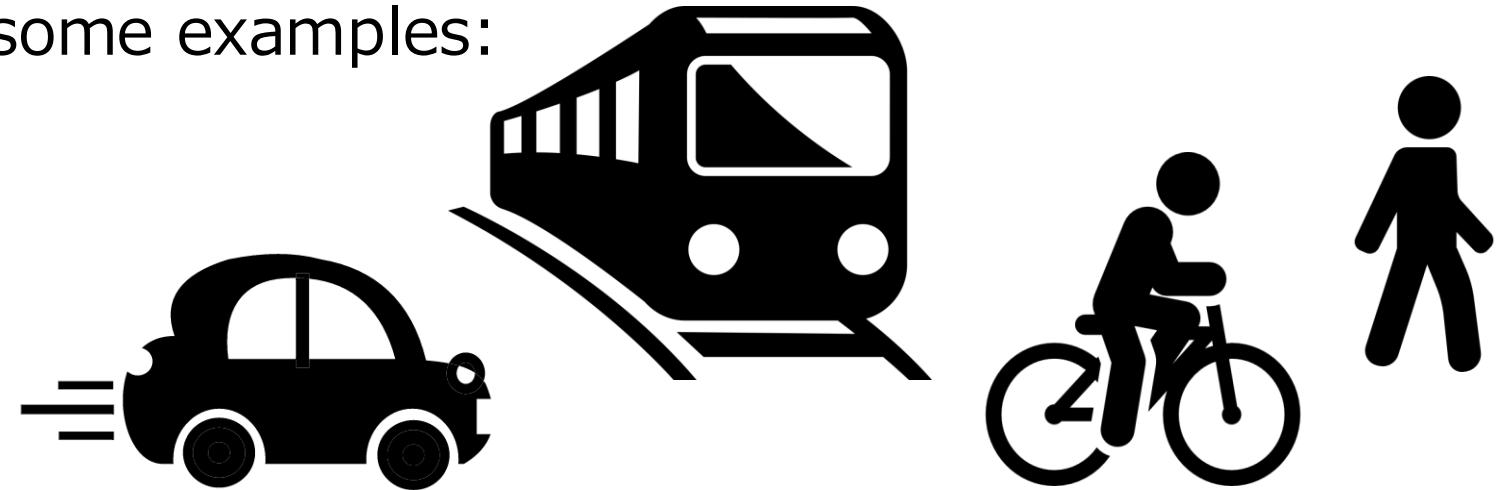
Persistent demonstrators

OGC delivers to the world a WEALTH of knowledge that goes way beyond the standards we're known for!

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

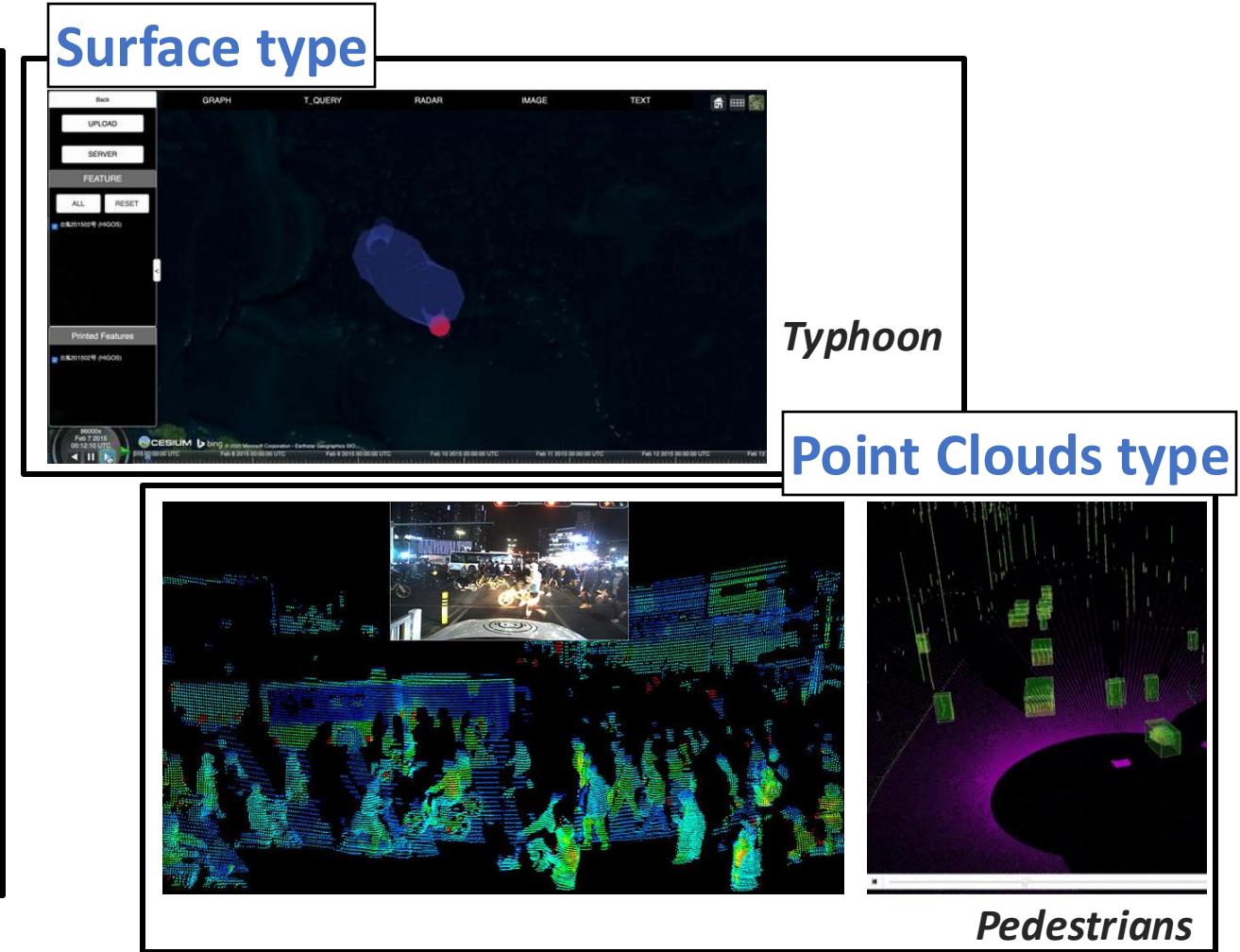
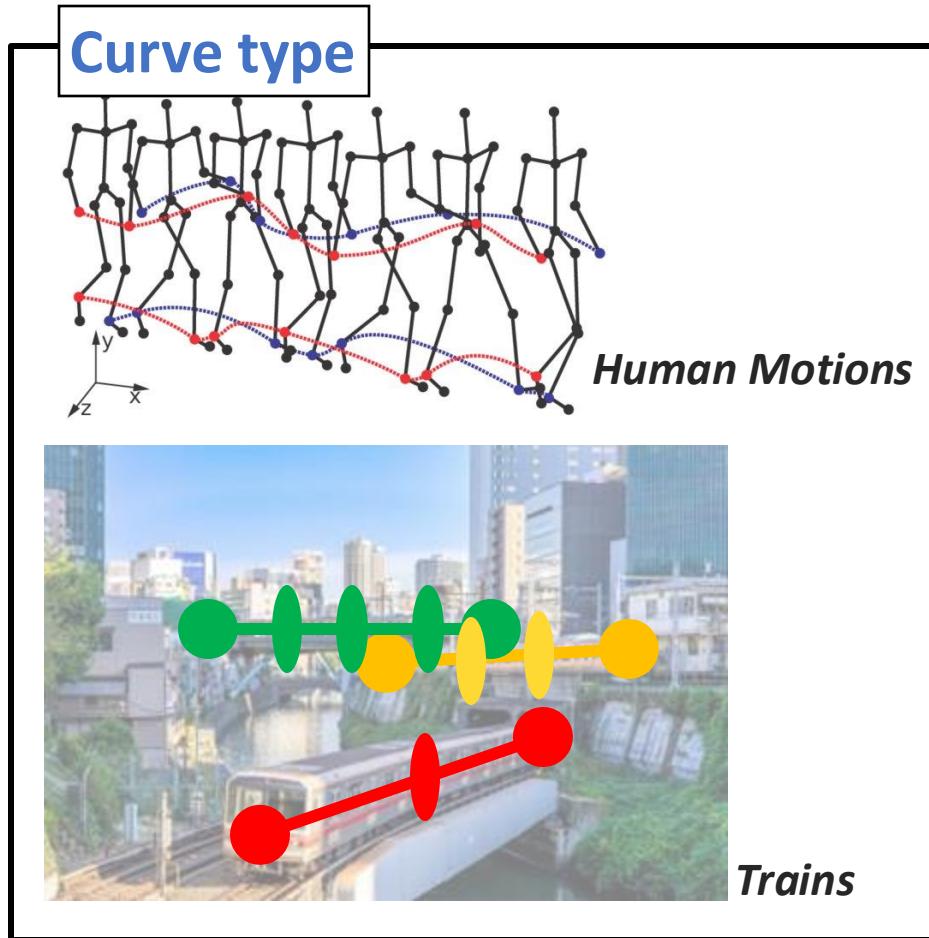


Image source:

1. <https://matcha-jp.com/en/4409>
2. <https://github.com/aisstairc/mf-cesium/wiki/Stinuum-Web-Manual>
3. <https://www.robosense.ai/en/tech-show-55>
4. <https://www.sama.com/3d-lidar-radar-point-cloud-annotation/>
5. 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.

OGC Moving Features SWG

Interoperability for spatiotemporal data and services
<https://www.ogc.org/standards/movingfeatures>

OGC Best Practice

OGC Standards

ISO Standards

Service Interface
Specifications

16-120r3 Moving Features Access
(guideline for implementing interfaces
to support moving feature data)

22-003r3 API – Moving Features – Part 1: Core
(for handling various moving feature data over HTTP)

Encoding
Specifications

**14-084r2 Simple
CSV** (compact
encoding for massive
moving points)

16-114r3 netCDF
(compact binary
encoding)

19-045r3 MF-JSON
(for encoding 0D, 1D, 2D, and 3D moving features with
dynamic non-spatial attributes)
as an OGC Standard Encoding

18-075 XML Core
(for encoding trajectories)

Data Model
(ISO 19141)

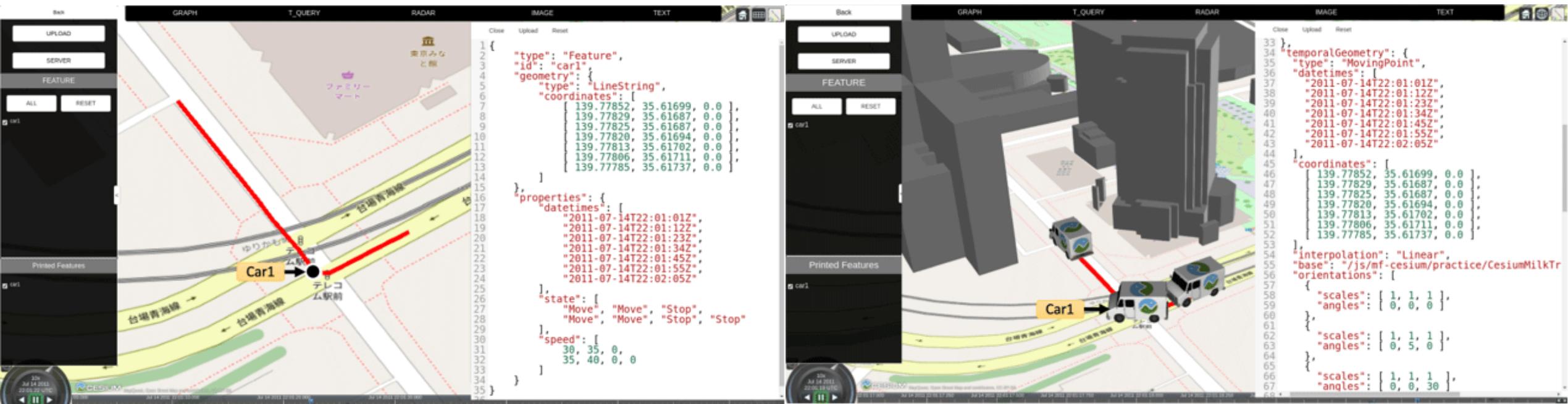
Moving Features 0D
(points)

Moving Features 1D/2D
(lines, curves, polygons,
etc.)

Moving Features 3D
(cubes, spheres, 3D model,
etc.)

OGC Moving Features JSON

- A new OGC standard for encoding and exchanging movement data of 2D and 3D objects



MF-JSON Trajectory

MF-JSON Prism

Github: <https://github.com/opengeospatial/mf-json>

Two encoding formats of MF-JSON

(a) MF-JSON Trajectory

```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "LineString",  
    "coordinates": [...,[...],...]  
  },  
  "properties": {  
    "datetimes": [...],  
    ....  
  },  
  "bbox": ...  
}
```

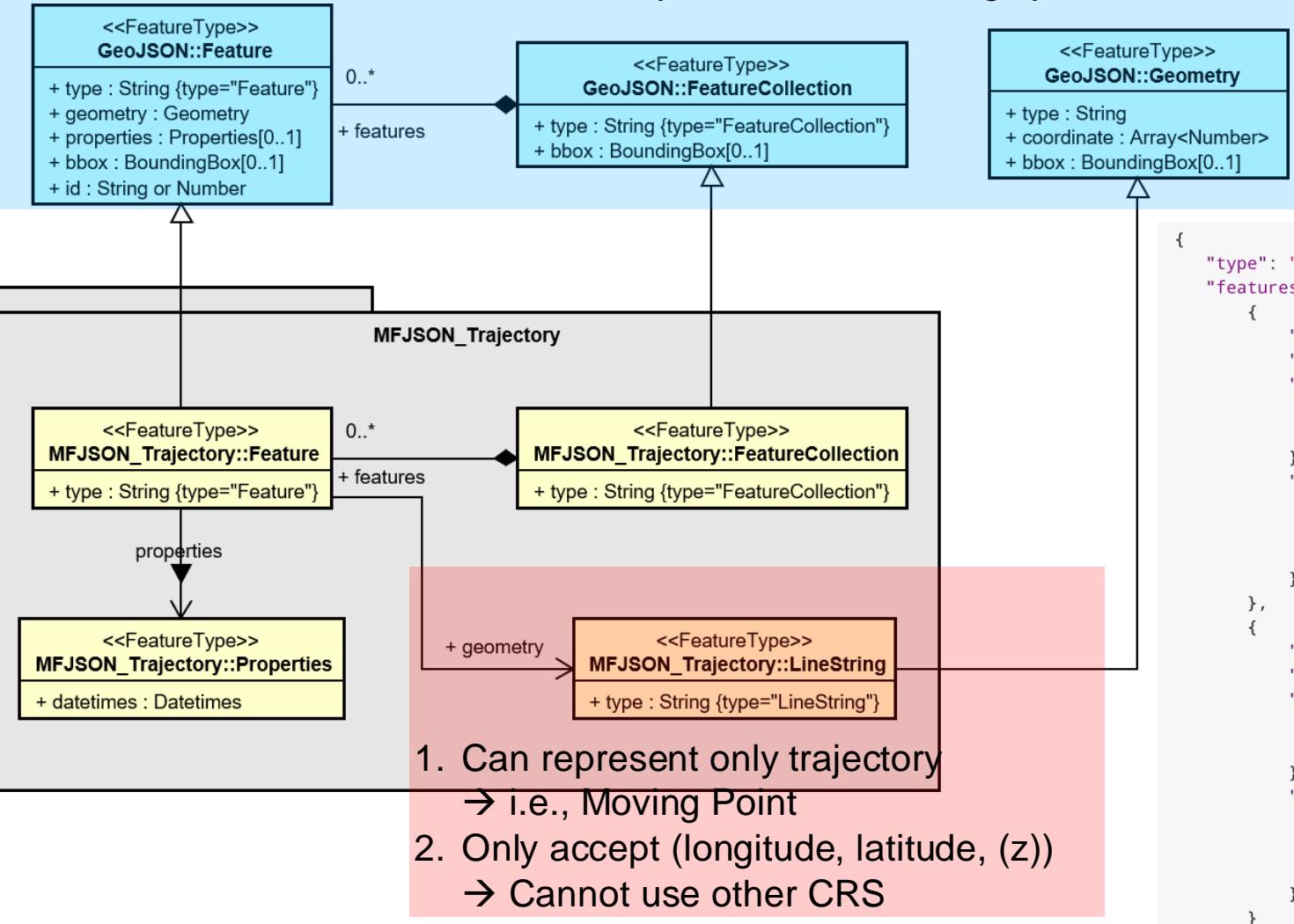
(b) MF-JSON Prism

```
{  
  "type": "Feature",  
  "geometry": ...,  
  "properties": ...,  
  "bbox": ..., New members  
  "crs": {...},  
  "trs":{...},  
  "temporalGeometry":{  
    "type":...,  
    "coordinates": [...],  
    "datetimes": [...],  
    ...  
  },  
  "temporalProperties": [...],  
  "time": [...]  
}
```

Image source: OGC Moving Features Encoding Extension JSON, <https://docs.ogc.org/is/19-045r3/19-045r3.html>

MF-JSON Trajectory

Inherit from GeoJSON → Easy to use with existing systems



JAVASCRIPT

```

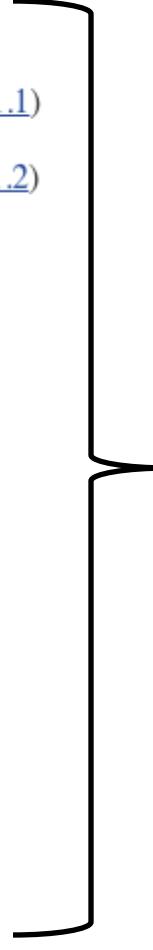
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "id": "A",
      "geometry": {
        "type": "LineString",
        "coordinates": [[11.0,2.0], [12.0,3.0], [10.0,3.0]]
      },
      "properties": {
        "datetimes": ["2012-01-17T12:33:51Z", "2012-01-17T12:33:56Z", "2012-01-17T12:34:00Z"],
        "state": ["walking","walking"],
        "typecode": [1, 2]
      }
    },
    {
      "type": "Feature",
      "id": "B",
      "geometry": {
        "type": "LineString",
        "coordinates": [[10.0,2.0], [11.0,3.0]]
      },
      "properties": {
        "datetimes": ["2012-01-17T12:33:51Z", "2012-01-17T12:34:00Z"],
        "state": ["walking"],
        "typecode": [2]
      }
    }
  ]
}
  
```

**With “datetimes”,
 Users can add a user-defined temporal property**

MF-JSON Prism

A MF-JSON Prism document may contain JSON objects that represent instances with the following types, as well as primitives types of JSON null, true, false, string, number, and array.

- TemporalGeometry object (see [7.2.1](#))
 - TemporalPrimitiveGeometry object (see [7.2.1.1](#))
 - TemporalComplexGeometry object (see [7.2.1.2](#))
- TemporalProperties object (see [7.2.2](#))
 - ParametricValues object (see [7.2.2.1](#))
- CoordinateReferenceSystem object (see [7.2.3](#))
- MovingFeature object (see [7.2.4](#))
- MovingFeatureCollection object (see [7.2.5](#))
- LifeSpan object (see [7.2.6](#))
- BoundingBox object (see [7.2.7](#))
- Geometry object (see [7.2.8](#))
- Properties object (see [7.2.9](#))
- MotionCurve object (see [7.2.10](#))



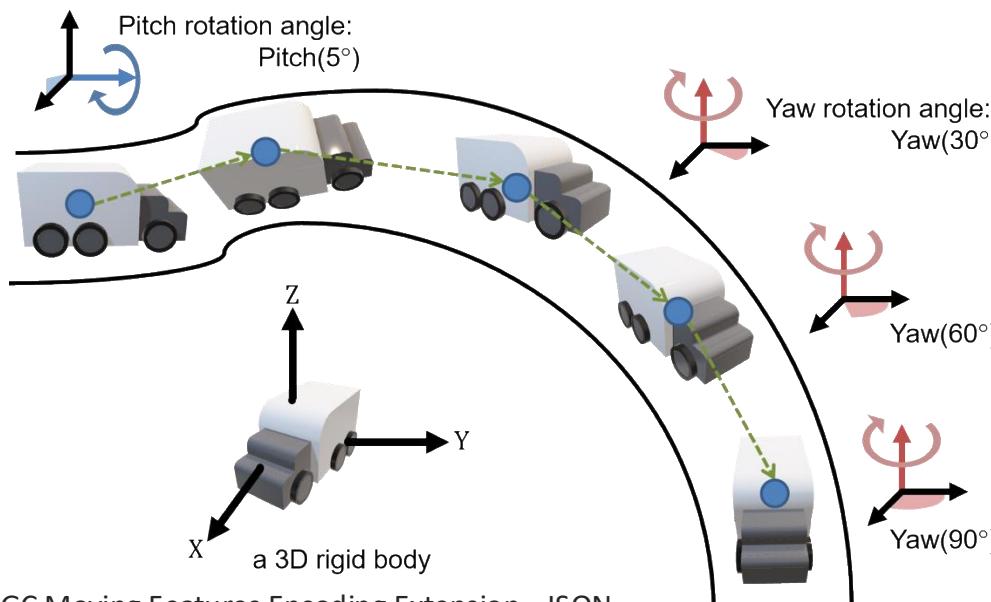
There are many objects for MF-JSON Prism

Key Features: TemporalGeometry

TemporalPrimitiveGeometry object

```
{
  // vbar | as a means to select ONE type.
  "type": "MovingPoint | MovingLineString | MovingPolygon | MovingPointCloud", // (MANDATORY)
  "datetimes": [...], // (MANDATORY)
  "coordinates": [...], // (MANDATORY)
  "interpolation": "...", // (DEFAULT)
  "base": {...}, // (OPTIONAL)
  "orientations": [...], // (OPTIONAL)
  "crs": {...}, // (DEFAULT)
  "trs": {...} // (DEFAULT)
}
```

Special elements for temporal geometry
(and 3D model)



TemporalComplexGeometry object

```
{
  "type": "MovingGeometryCollection", // (MANDATORY)
  "prisms": [ // (MANDATORY)
    {
      "type": "MovingPoint | MovingLineString | MovingPolygon | MovingPointCloud",
      means to select ONE type.
      "datetimes": [...], // (MANDATORY)
      "coordinates": [...], // (MANDATORY)
      "interpolation": "...", // (DEFAULT)
      ...
    }, // Array element: TemporalPrimitiveGeometry object
    "crs": {...}, // (DEFAULT)
    "trs": {...} // (DEFAULT)
  ]
}
```

Image source: OGC Moving Features Encoding Extension - JSON,
<https://docs.ogc.org/is/19-045r3/19-045r3.html>

Key Features: Interpolation of Movement

Predefined MotionCurve object

```
{  
    ...,  
    "datetimes": [...], //T={t_0, t_1, ..., t_n}  
    "coordinates": [...], //G={G_0, G_1, ..., G_n}  
    "interpolation": "Discrete|Step|Linear|Quadratic|Cubic", // vbar | as a means to select ONE.  
}  
...
```

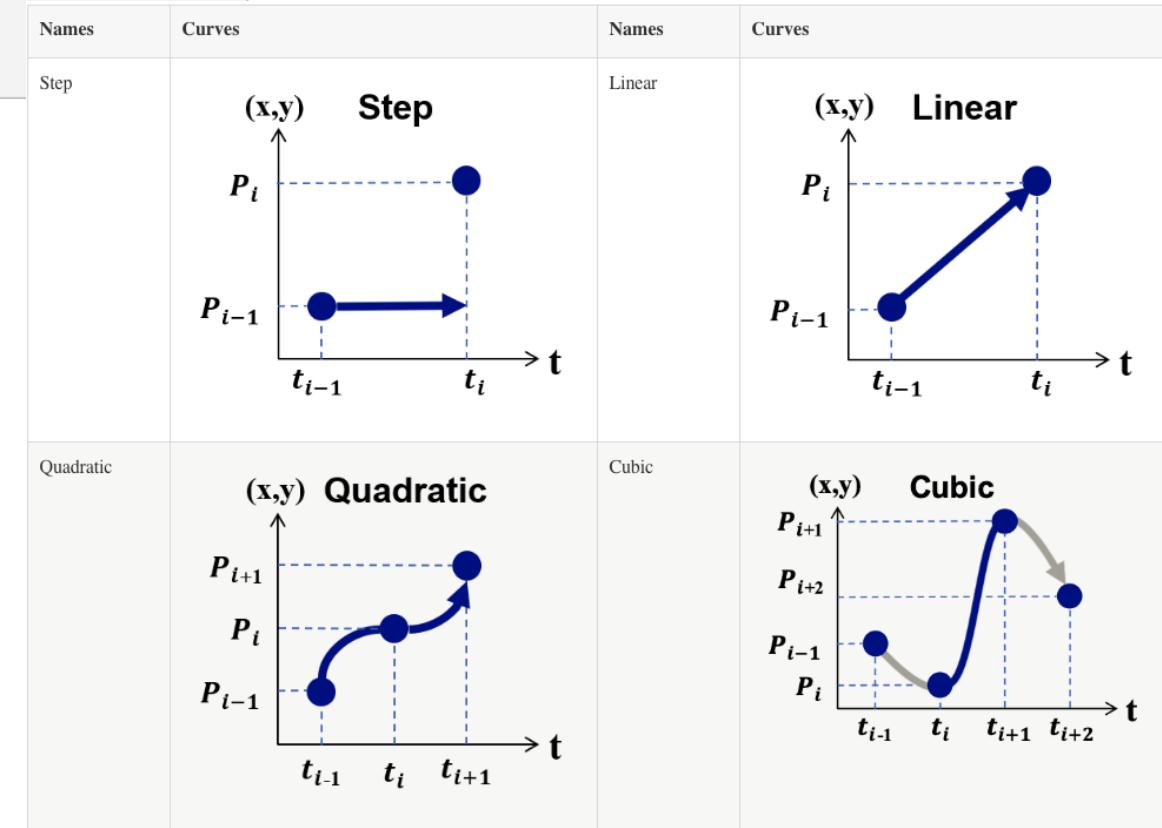


Image source: OGC Moving Features Encoding Extension JSON, <https://docs.ogc.org/is/19-045r3/19-045r3.html>

Key Features: TemporalProperties

ParametricValues object

<pre>{ "datetimes": [...], // (MANDATORY) a JSON Array of time instants "@property0": { // @property0 whose name is any string defined by an application. "type": "Measure", // (MANDATORY) a predefined string among 'Measure', 'Text', and 'Image' "values": [...], // (MANDATORY) a JSON Array of values "interpolation": "...", // (DEFAULT) a predefined string or a URL "form": "...", // (OPTIONAL) a unit of measurement "description": "any string" // (OPTIONAL) any string content for an application }, "@property1": { // @property1 whose name is any string defined by an application, but the name is not the same as "@property0". "type": "Text", // @property1 has values at the same time instants of @property0 ... }, "@property2" : { // @property1 whose name is any string defined by an application, but "@property0" nor @property1. "type": "Image", } }</pre>	<p>Shared time information</p> <p>Temporal property block</p> <p>Predefined temporal property type</p>																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Type strings</th> <th style="text-align: left; padding: 5px;">Descriptions</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">Measure</td> <td style="padding: 5px;">The "values" member contains any numeric values.</td> </tr> <tr> <td style="padding: 5px;">Text</td> <td style="padding: 5px;">The "values" member contains any strings.</td> </tr> <tr> <td style="padding: 5px;">Image</td> <td style="padding: 5px;">The "values" member contains Base64 strings converted from images or URLs to address images.</td> </tr> </tbody> </table>	Type strings	Descriptions	Measure	The "values" member contains any numeric values.	Text	The "values" member contains any strings.	Image	The "values" member contains Base64 strings converted from images or URLs to address images.	<p>Predefined interpolation for temporal property</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Names</th> <th style="text-align: left; padding: 5px;">Descriptions</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">Discrete</td> <td style="padding: 5px;">The sampling of the attribute occurs such that it is not possible to regard the series as continuous; thus, there is no interpolated value if t is not an element in "datetimes".</td> </tr> <tr> <td style="padding: 5px;">Step</td> <td style="padding: 5px;">The values are not connected at the end of a subinterval with two successive instants. The value just jumps from one value to the other at the end of a subinterval.</td> </tr> <tr> <td style="padding: 5px;">Linear</td> <td style="padding: 5px;">The values are essentially connected and a linear interpolation estimates the value of the property at the indicated instant during a subinterval.</td> </tr> <tr> <td style="padding: 5px;">Regression</td> <td style="padding: 5px;">The value of the attribute at the indicated instant is extrapolated from a simple linear regression model with the whole values corresponding to the all elements in "datetimes".</td> </tr> </tbody> </table>	Names	Descriptions	Discrete	The sampling of the attribute occurs such that it is not possible to regard the series as continuous; thus, there is no interpolated value if t is not an element in "datetimes".	Step	The values are not connected at the end of a subinterval with two successive instants. The value just jumps from one value to the other at the end of a subinterval.	Linear	The values are essentially connected and a linear interpolation estimates the value of the property at the indicated instant during a subinterval.	Regression	The value of the attribute at the indicated instant is extrapolated from a simple linear regression model with the whole values corresponding to the all elements in "datetimes".
Type strings	Descriptions																		
Measure	The "values" member contains any numeric values.																		
Text	The "values" member contains any strings.																		
Image	The "values" member contains Base64 strings converted from images or URLs to address images.																		
Names	Descriptions																		
Discrete	The sampling of the attribute occurs such that it is not possible to regard the series as continuous; thus, there is no interpolated value if t is not an element in "datetimes".																		
Step	The values are not connected at the end of a subinterval with two successive instants. The value just jumps from one value to the other at the end of a subinterval.																		
Linear	The values are essentially connected and a linear interpolation estimates the value of the property at the indicated instant during a subinterval.																		
Regression	The value of the attribute at the indicated instant is extrapolated from a simple linear regression model with the whole values corresponding to the all elements in "datetimes".																		

Image source: OGC Moving Features Encoding Extension - JSON,
<https://docs.ogc.org/is/19-045r3/19-045r3.html>

Example of TemporalProperties object

```
"temporalProperties": [ // (OPTIONAL) dynamic non-spatial attributes, extended from 'properties'  
  { // a group of temporal properties that are measured at the same times  
    "datetimes": ["2011-07-14T22:01:01.450Z", "2011-07-14T23:01:01.450Z", "2011-07-15T00:01:01.450Z"],  
    "length": {  
      "type": "Measure",  
      "form": "http://www.qudt.org/qudt/owl/1.0.0/quantity/Length", // a URI denoting a unit-of-measure  
      "values": [1.0, 2.4, 1.0], // the array of values for "length", with the same number of elements as "datetimes"  
      "interpolation": "Linear",  
      "description": "description1" // (OPTIONAL)  
    },  
    "discharge" : {  
      "type" : "Measure",  
      "form" : "MQS", // a symbol for m^3/s(a cubic metre per second)  
      "values" : [3.0, 4.0, 5.0],  
      "interpolation": "Step"  
    }  
  }, {  
    "datetimes" : [1465621816590, 1465711526300],  
    "camera" : {  
      "type" : "Image",  
      "values" : ["http://www.opengis.net/spec/movingfeatures/json/1.0/prism/example/image1", "iVBORw0KGgoAAAANSUhEU....."],  
      "interpolation": "Discrete"  
    },  
    "labels":{  
      "type": "Text", // a predefined unit for a string value  
      "values": ["car", "human"], // the array of values for "labels", with the same number of elements as "datetimes"  
      "interpolation": "Discrete",  
      "description": "description2" // (OPTIONAL)  
    }  
  }  
]
```

Image source: OGC Moving Features Encoding Extension - JSON,
<https://docs.ogc.org/is/19-045r3/19-045r3.html>

Key Features: MovingFeature

{

Inherited from GeoJSON

- “type”: (Mandatory), “Feature”, the same semantics of GeoJSON
- “id”: (Optional), the same semantics of GeoJSON
- “geometry”: (Optional), the same semantics of GeoJSON
- “properties”: (Optional), the same semantics of GeoJSON
- “bbox”: (Optional), the same semantics of GeoJSON
- “time”: (Optional), the temporal coordinate range for MovingFeature
- “crs”: (Default: WGS84), the spatial coordinate reference system
- “trs”: (Default: ISO8601), the temporal coordinate reference system
- “temporalGeometry”: (Mandatory), the dynamic spatial attributes
- “temporalProperties”: (Optional), the dynamic non-spatial attributes

}

Additional properties for MF-JSON

Summary of OGC MF–JSON Encoding

- <https://docs.opengeospatial.org/is/19-045r3/19-045r3.html>
- OGC MF–JSON encoding covered various cases
 - **Core:** MovingFeature (and MovingFeatureCollection) Object
 - **Location:** TemporalGeometry Object
 - Types: **MovingPoint**, **MovingLineString**, **MovingPolygon**, and **MovingPointCloud**
 - Predefined interpolations: **Discrete**, **Step**, **Linear**, **Quadratic**, and **Cubic**
 - **Properties:** TemporalProperties Object
 - Types: **Measure**, **Text**, and **Image**
 - Predefined interpolations: **Discrete**, **Step**, **Linear**, and **Regression**
- There are other conceptual (and encoding) models for moving features, such as OGC MF XML, CSV, netCDF, and so on;
 - But they have not fully covered various cases.

OGC API–Moving Features consists of four parts

- **Part 1: Core (Features Extension)**

- Moving feature is also a kind of feature.
 - Inherit the necessary APIs (and Resources) from OGC API – Features (& Common)
- Consider **CRUD operations for *MF-JSON* objects**
 - CRUD: Create, Read, Update, and Delete
- **Support fundamental operations**
 - Based on OGC Moving Features Access **Type A** operations

- **Part 2: Filtering Extension**

- Based on OGC Moving Features Access **Type B&C** operations (**Synchronous**)

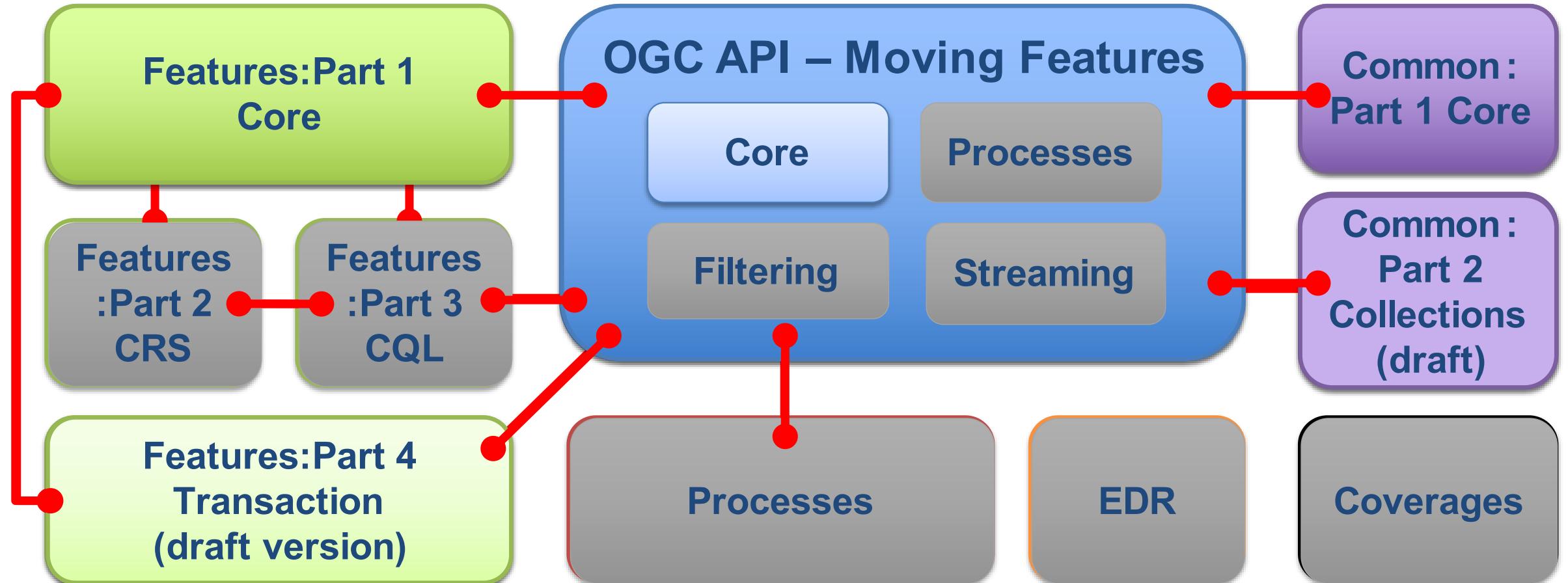
- **Part 3: Processes Extension**

- Based on OGC Moving Features Access **Type B&C** operations (**Asynchronous**)

- **Part 4: Streaming Extension**

- Consider real-time moving features

Relationship with other OGC APIs



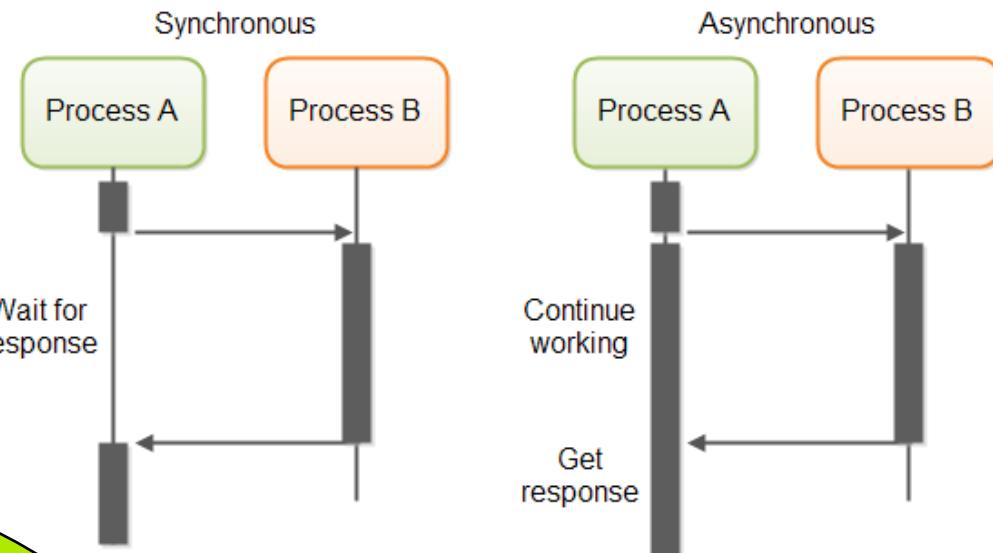
Start from the synchronous way

OGC API – MF Part 3: **Processes** Extension (Asynchronous)

OGC API – MF Part 2: **Filtering** Extension
(Synchronous)

OGC API – MF Part 1: **Core** (Synchronous)

OGC MF-JSON encoding



OGC MF – API Part 4:
Streaming Extension

OGC API – Moving Features – Part 1: Core

- ***OGC API – Moving Features – Part 1: Core is published!***

- **Overview:**

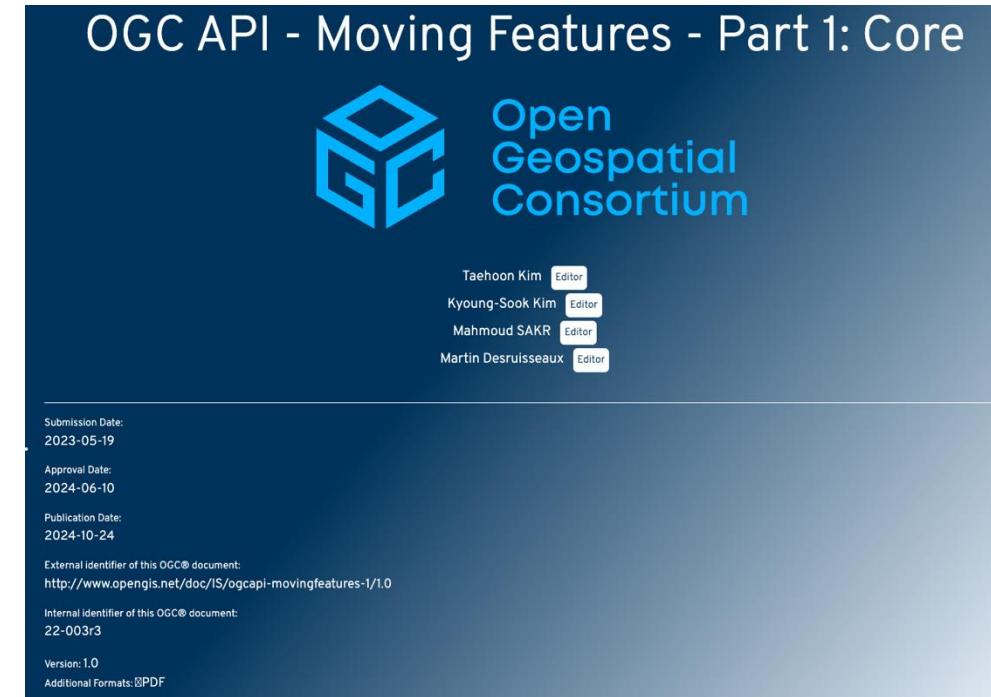
- <https://ogcapi.ogc.org/movingfeatures/>
- <https://github.com/opengeospatial/ogcapi-movingfeatures>

- **Document:**

- <https://docs.ogc.org/is/22-003r3/22-003r3.html>

- **OpenAPI:**

- <https://developer.ogc.org/api/movingfeatures/>



- 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**.

Overview of OGC API – MF – Part 1:Core

Inherited from
OGC API-Features

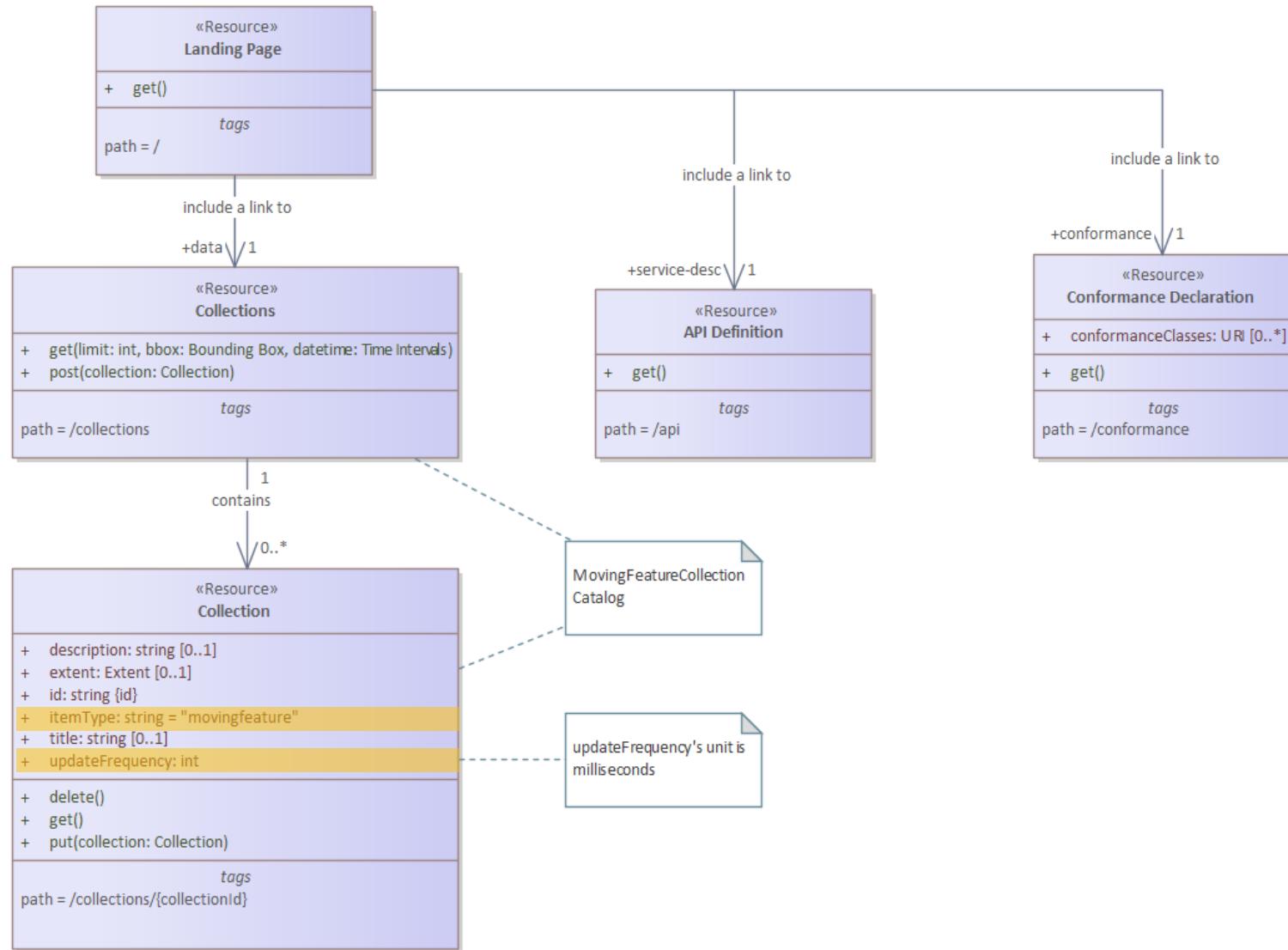
Resources from
OGC MF-JSON

Resources from
OGC MF-Access

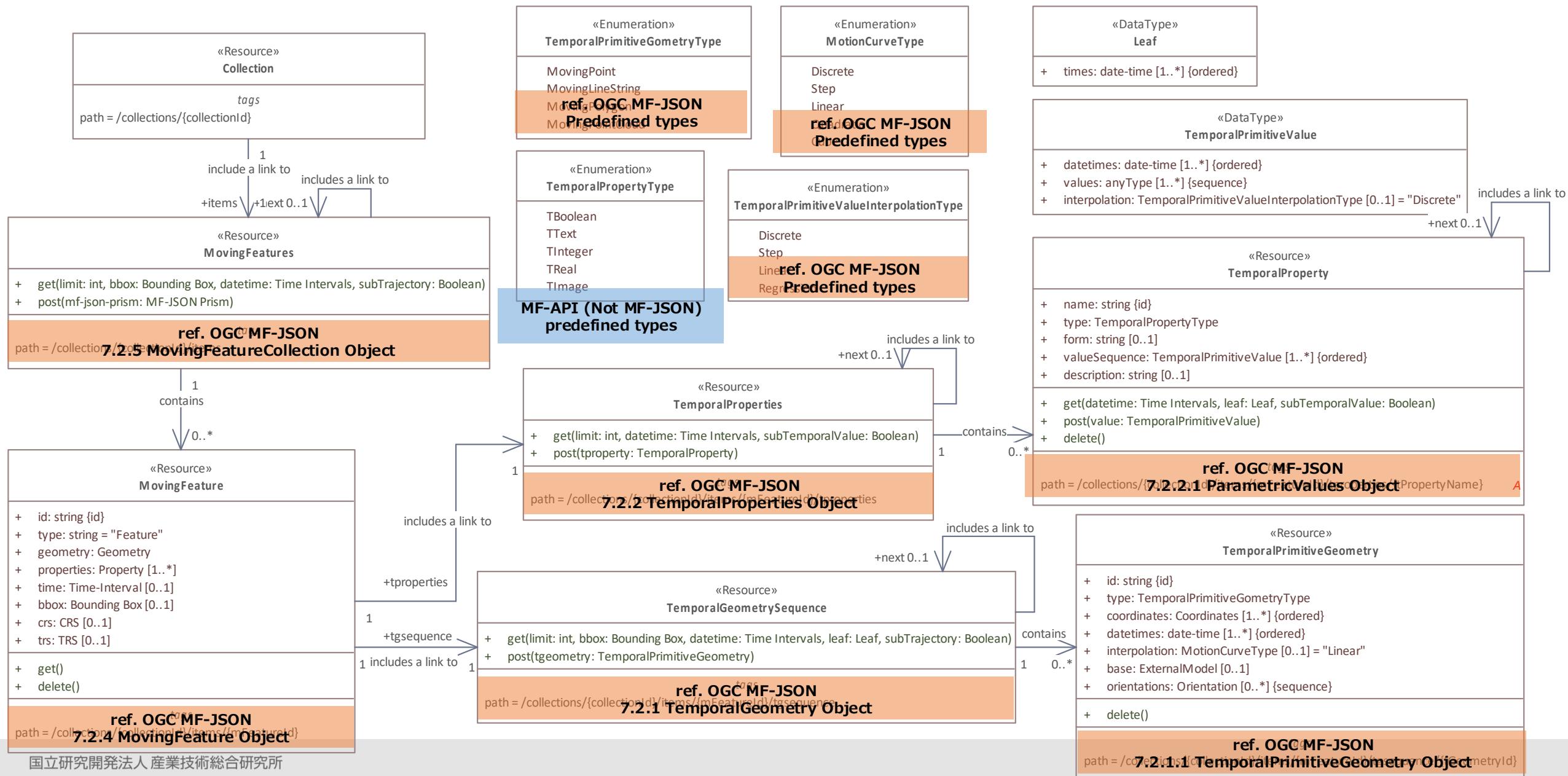
Resource	Path	HTTP Method	Document Reference
Landing page	/	GET	No modification
API definition	/api	GET	
Conformance classes	/conformance	GET	
Collections metadata	/collections	GET, POST	7.3
Collection instance metadata	/collections/{collectionId}	GET, DELETE, PUT	7.4
Moving feature collection	/collections/{collectionId}/items	GET, POST	8.3
Moving feature instance	/collections/{collectionId}/items/{mFeatureId}	GET, DELETE	8.4
Temporal geometry sequence	/collections/{collectionId}/items/{mFeatureId}/tgsequence	GET, POST	8.5
Temporal primitive geometry	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}	DELETE	8.6
Temporal properties	/collections/{collectionId}/items/{mFeatureId}/tproperties	GET, POST	8.8
Temporal property instance	/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}	GET, POST, DELETE	8.9
TemporalPrimitiveValue instance	/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}/{tValueId}	DELETE	8.10
Velocity query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/velocity	GET	8.7
Acceleration query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/acceleration	GET	8.7
Distance query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/distance	GET	8.7

Inherited API – Features and Common

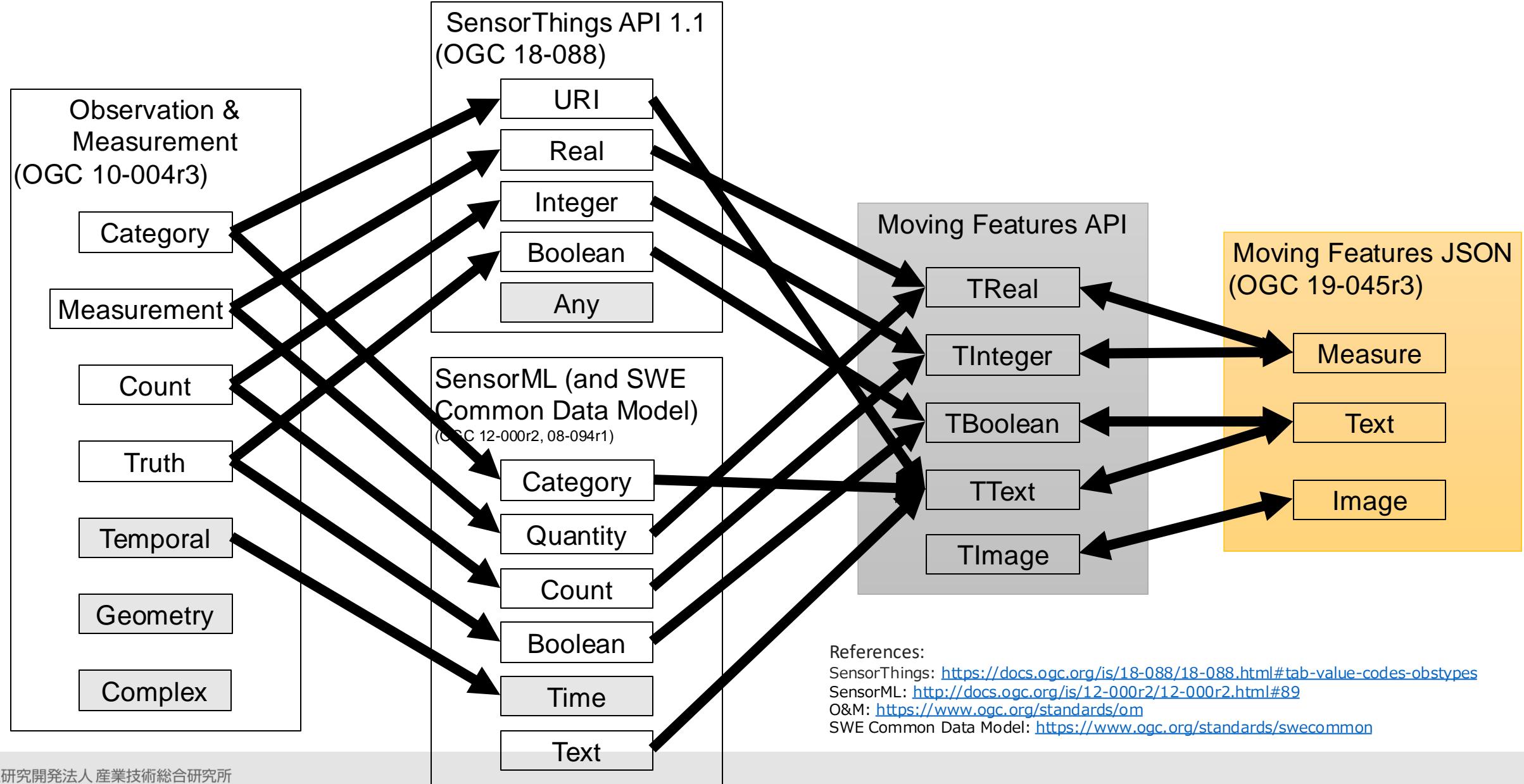
- There is **no modification** to support moving features.
- But there are two constraints:
 - The ***itemType*** of Collection should be "**movingfeature**"
 - The new property **(*updateFrequency*)** is required to determine the continuous of temporal geometry
 - Update frequency: a time interval of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



Resource diagram of OGC API – MF



Predefined Temporal Property Types



Relationship of Between Web Resources

- ***TemporalGeometry*** is a kind of ***TemporalProperty***
- ***TemporalPrimitiveGeometry*** is a kind of ***TemporalPrimitiveValue***

MovingFeatures (~MovingFeatureCollection)

MovingFeature A

TemporalGeometry

TemporalPrimitiveGeometry 1



TemporalPrimitiveGeometry N

TemporalProperties

TemporalProperty A

TemporalPrimitiveValue 1



TemporalPrimitiveValue M

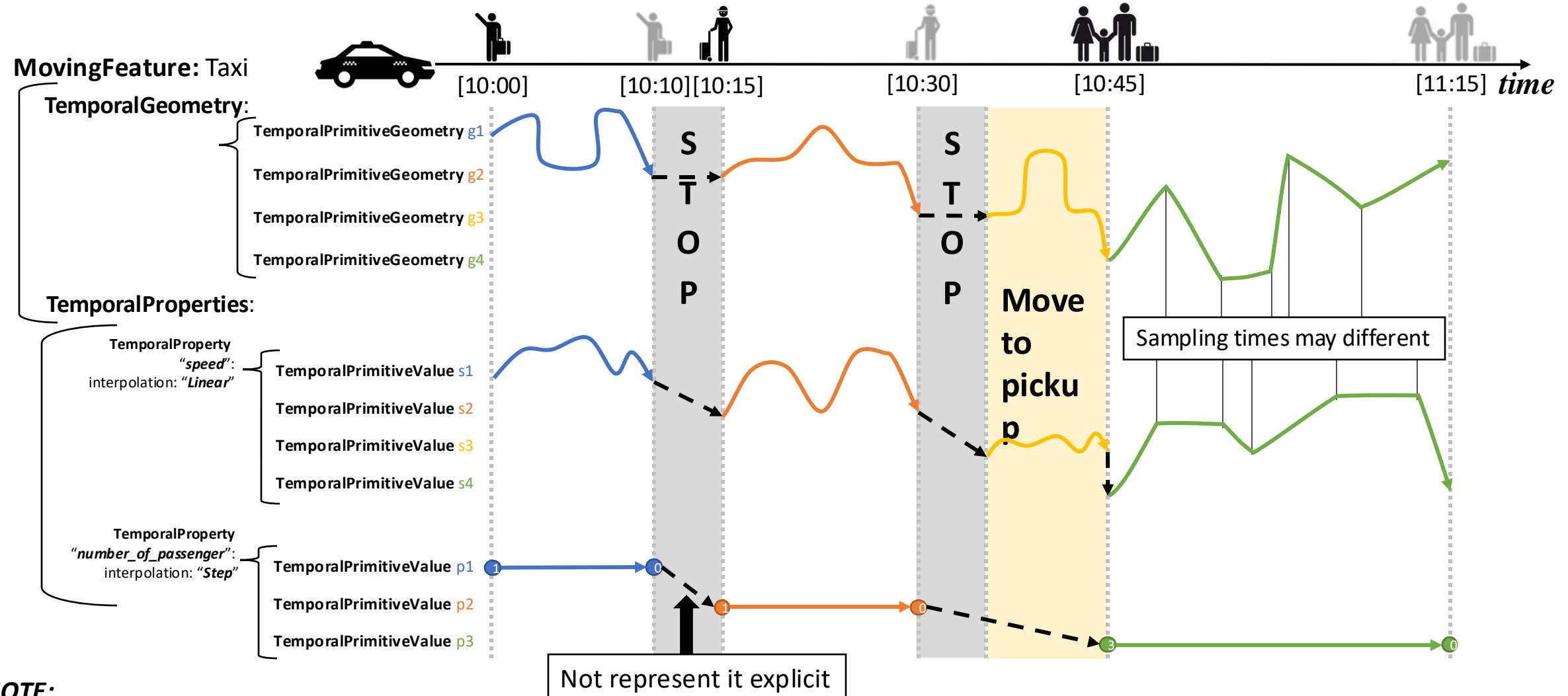


TemporalProperty B

Moving
Feature B



Example of Moving Feature



NOTE:

TemporalGeometry is a kind of **TemporalProperty**

TemporalPrimitiveGeometry is a kind of **TemporalPrimitiveValue**

Operations Defined in OGC MF Access

<<Type>>
TemporalTrajectory

```
+ pointAtTime(TemporalCoordinate): DirectPosition
+ timeAtPoint(DirectPosition): Set<TemporalGeometricPrimitive>
+ velocityAtTime(TemporalCoordinate): Velocity
+ accelerationAtTime(TemporalCoordinate): Acceleration
+ timeAtDistance(Distance): TemporalGeometricPrimitive[0..*]
+ cummulativeDistanceAtTime(TemporalCoordinate): Distance
+ timeToDistance(): Curve[1..*]
+ timeAtCummulativeDistance(Distance): TemporalGeometricPrimitive[0..*]
+ subTrajectory(): TemporalTrajectory
+ positionAtTime(): LinearReferencePositionExpression
+ snapToGrid(Point, Distance[]): TemporalTrajectory
```

Covered by Part 1 Features Extension
<Relatively simple functions>

Type A:
Retrieval of
feature attribute

```
+ equals(Point, TemporalPeriod): Boolean
+ disjoint(Geometry, TemporalPeriod): Boolean
+ intersects(Geometry, TemporalPeriod): Boolean
+ distanceWithin(Geometry, TemporalPeriod, Distance): Boolean
+ intersection(Geometry, TemporalPeriod): TemporalTrajectory
+ difference(Geometry, TemporalPeriod): TemporalTrajectory
+ nearestApproach(Geometry, TemporalPeriod): Distance, TemporalGeometricPrimitive[1..*]
+ nearestApproachPoint(Geometry, TemporalPeriod): directPosition, TemporalGeometricPrimitive[1..*]
+ equals(TemporalTrajectory, TemporalPeriod): Boolean
+ disjoint(TemporalTrajectory, TemporalPeriod): Boolean
+ intersects(TemporalTrajectory, TemporalPeriod): Boolean
+ distanceWithin(TemporalTrajectory, TemporalPeriod, Distance): Boolean
+ durationWithin(TemporalTrajectory, TemporalPeriod, TemporalDuration): Boolean
+ intersection(TemporalTrajectory, TemporalPeriod): Set<DirectPosition>
+ nearestApproach(TemporalTrajectory, TemporalPeriod): Distance, GeometricPrimitive[1..*]
+ nearestApproachPoint(TemporalTrajectory): directPosition, GeometricPrimitive[1..*]
```

Covered by Part 2 Filtering Extension
(& Part 3 Processes Extension)

<Relatively complex functions>

Type B:
Operations between
one trajectory object
and one or more
geometry objects

Type C:
Operations between
two trajectory objects

Type A Operation List

- pointAtTime
- velocityAtTime
- accelerationAtTime
- **cummulativeDistanceAtTime**
- ~~positionAtTime~~
- ~~timeAtPoint~~
- **timeAtDistance**
- **timeToDistance**
- subTrajectory
- ~~snapToGrid~~

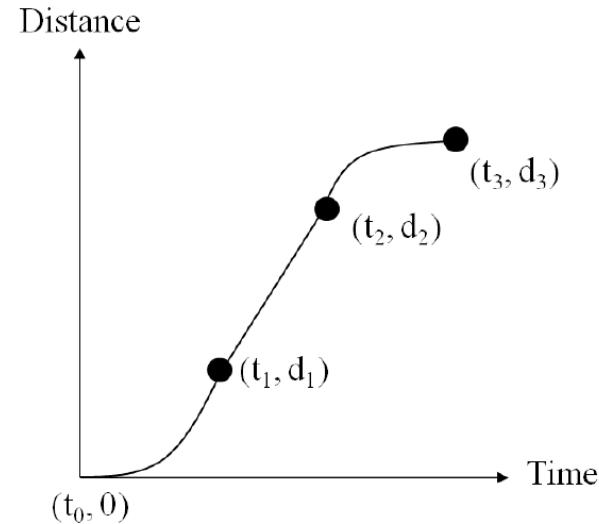


Figure 8 — Example of time to distance curve

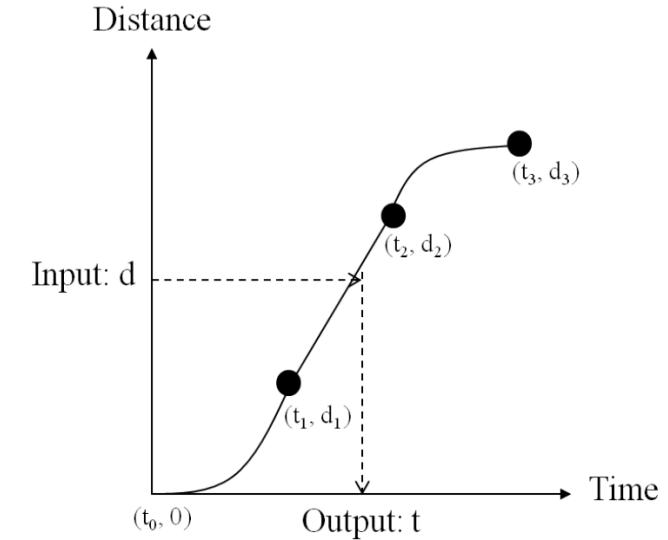


Figure 9 — Example of timeAtDistance

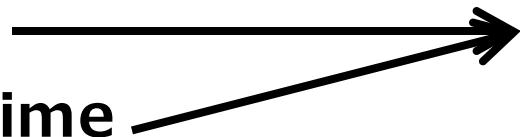
These operations can get results simply from the time-to-distance curve

Define Distance Resource

- a kind of TemporalProperty Resource to represent a time-to-distance curve
- derived from TemporalPrimitiveGeometry Resource

Type A Operation List

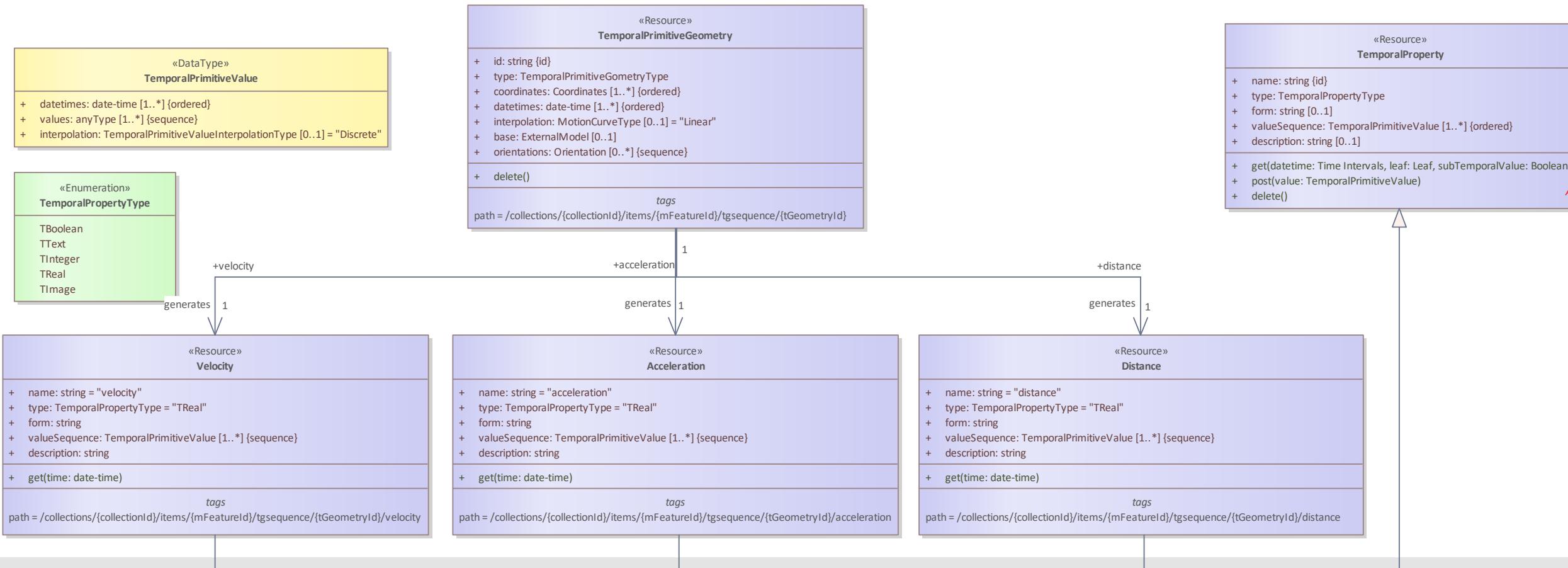
- pointAtTime
- **velocityAtTime**
- **accelerationAtTime**
- cummulativeDistanceAtTime
- ~~positionAtTime~~
- ~~timeAtPoint~~
- timeAtDistance
- timeToDistance
- subTrajectory
- ~~snapToGrid~~



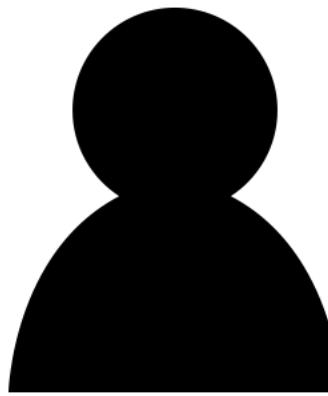
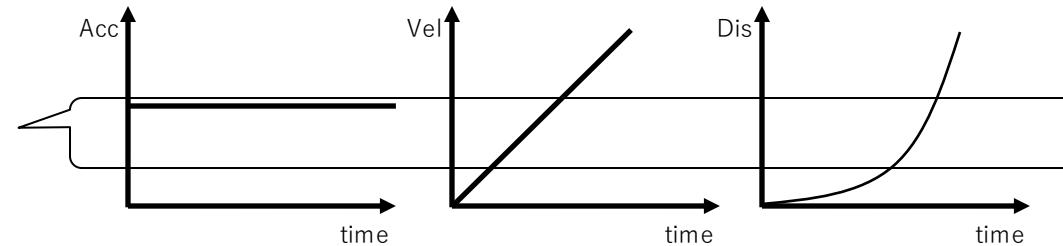
Similarly, can be supported by defining Velocity (and Acceleration) Resource like the Distance resource

Resources for Temporal Geometry Query

- **Distance, Velocity, and Acceleration** resources support only HTTP GET operation
 - A kind of **TemporalProperty** with a specified type of value
 - The result schema is **TemporalProperties** in **MF-JSON**

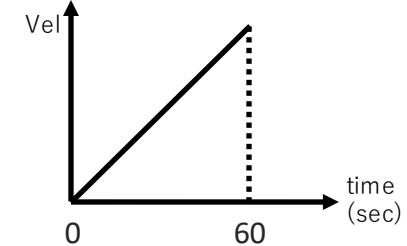


Examples with Queries



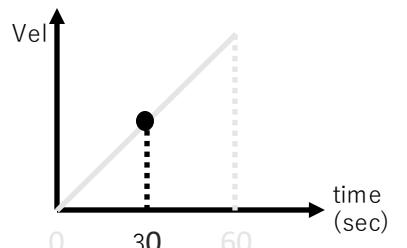
```
GET
/collections/col-1/items/mf-1
/tgsequence/tg-1/velocity
```

```
{
  "temporalproperties": [
    {
      "datetimes": [
        "2022-11-01T10:00:00Z",
        "2022-11-01T10:01:00Z"
      ],
      "velocity": {
        "type": "Measure",
        "form": "KMH",
        "values": [0.0, 100.0],
        "interpolation": "Linear"
      }
    }
  ]
}
```



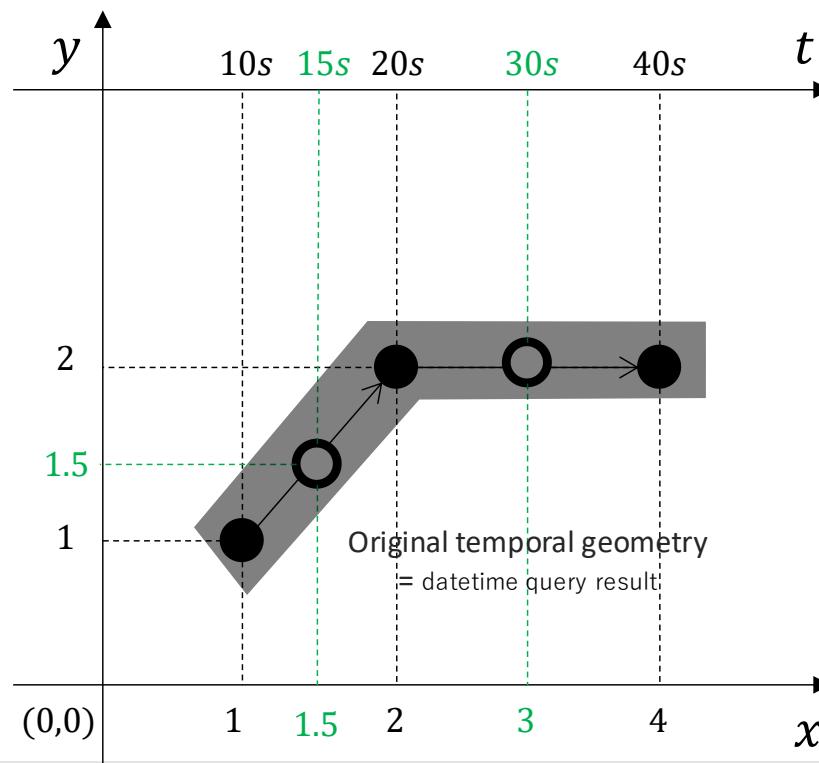
```
GET
/collections/col-1/items/mf-1
/tgsequence/tg-1/velocity
?datetime=2022-11-01T10:00:00/..
&leaf=2022-11-01T10:00:30
```

```
{
  "temporalproperties": [
    {
      "datetimes": [
        "2022-11-01T10:00:30Z"
      ],
      "velocity": {
        "type": "Measure",
        "form": "KMH",
        "values": [50.0],
        "interpolation": "Discrete"
      }
    }
  ]
}
```



datetime Parameter

- The parameter 'datetime' is defined in the OGC API – Features (and Common)
 - 'datetime' is a date-time string.
 - a time instance, a half-bounded time interval, and a bounded time interval
 - It returns the features intersected over time as its result.
 - This is a required parameter in the API – MF.



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

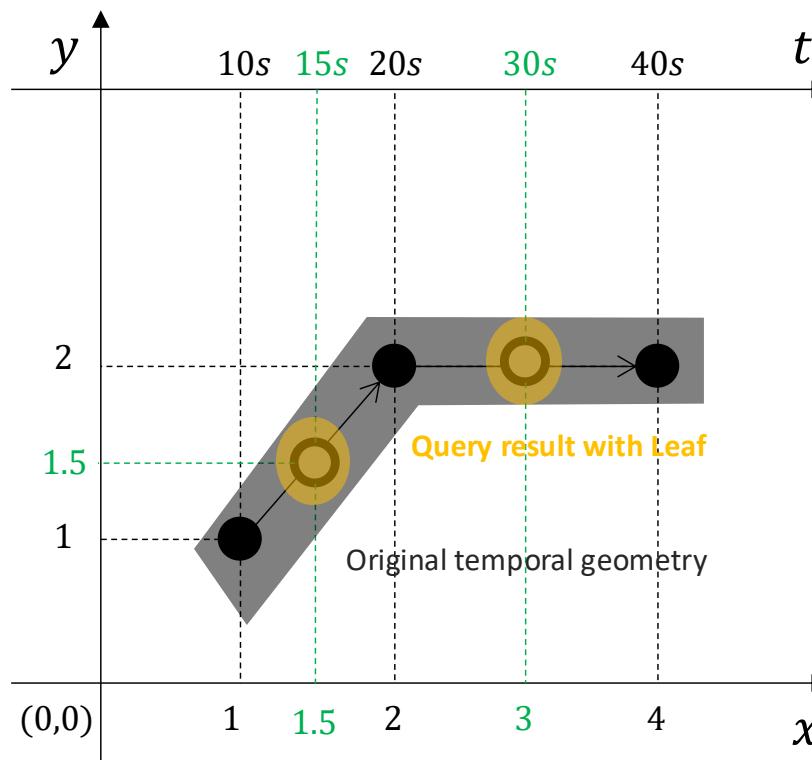
<temporal primitive geometry object> <tgsequencequery response>

```
{
  "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",
}

...
"geometrySequence": [
{
  "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",
},
...
```

leaf Parameter

- The parameter 'leaf' can be used in the HTTP GET operation of **TemporalGeometrySequence** and **TemporalProperty**
 - 'leaf' is a sequence of date-time strings (i.e., an array of date-time strings).
 - Implements ***pointAtTime*** operation in the [OGC Moving Feature Access standard](#)



```
<temporal primitive 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",
}
```

```
<tsequencequery response>
```

```

<temporal primitive geometry object>
  <geometrySequence>
    {
      "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",
    }
  ...

```

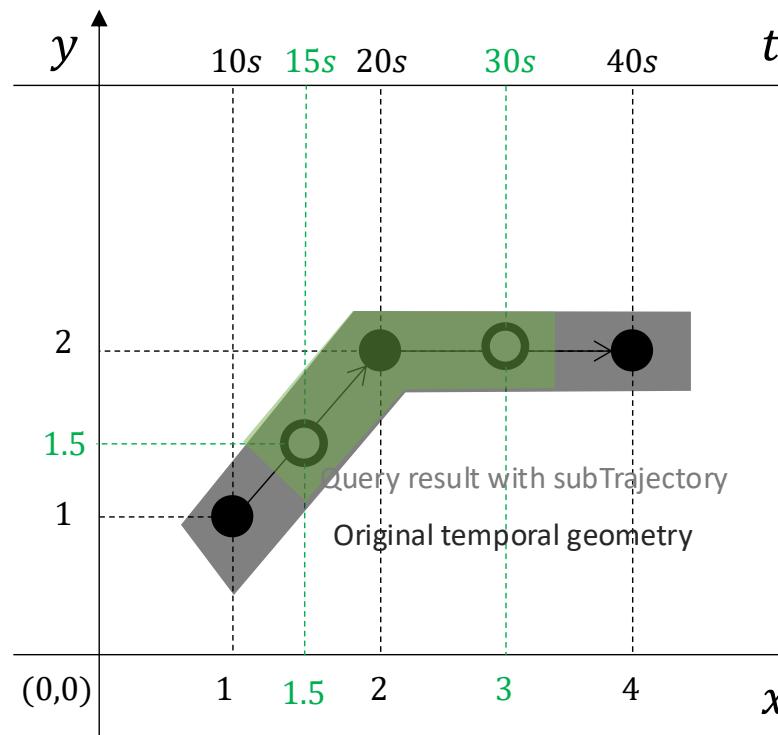
```

<tsequencequery response>
  <geometrySequence>
    {
      "id": "tgeom",
      "type": "MovingPoint",
      "datetimes": [
        "2021-09-14T12:00:15Z",
        "2021-09-14T12:00:30Z"
      ],
      "coordinates": [
        [1.5,1.5],
        [3,2]
      ],
      "interpolation": "Discrete",
    }
  ...

```

subTrajectory (and subTemporalValue) Parameter

- The parameter 'subTrajectory' can be used in the HTTP GET operation of **MovingFeatures** and **TemporalGeometrySequence**
 - 'subTrajectory' has a Boolean. A specified time interval is derived from the 'datetime' parameter.
 - Implements **subTrajectory** operation in the [OGC Moving Feature Access standard](#).
 - Similarly, for **TemporalProperties**, there is a 'subTemporalValue' parameter.



<temporal primitive 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

<itemsquery response>

```
...  
  "temporalGeometry":  
  {  
    "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 subTrajectory = true
and datetime=
"2021-09-14T12%3A00%3A15Z%2F
2021-09-14T12%3A00%3A30Z", then

<itemsquery response>

```
...  
  "temporalGeometry":  
  {  
    "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",  
  },  
  ...
```

Example GET /collections response

```
{
  "collections": [
    {
      "id": "mfc-1",
      "title": "MovingFeatureCollection_1",
      "description": "a collection of moving features to manage data in a distinct (physical or logical) space",
      "itemType": "movingfeature",
      "updateFrequency": 1000,
      "extent": {
        "spatial": {
          "bbox": [
            -180, -90, 190, 90
          ],
          "crs": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        },
        "temporal": {
          "interval": [
            "2011-11-11T12:22:11Z", "2012-11-24T12:32:43Z"
          ],
          "trs": "http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"
        }
      },
      "links": [
        {
          "href": "https://data.example.org/collections/mfc-1",
          "rel": "self",
          "type": "application/json"
        }
      ]
    }
  ],
  "links": [
    {
      "href": "https://data.example.org/collections",
      "rel": "self",
      "type": "application/json"
    }
  ]
}
```

Special member for MovingFeature

We can get FeatureCollection in the Server
(Collection of the MovingFeatureCollection)
 This work is mainly supported by OGC API - Features.

Collection Resource Instance
 → GET /collections/mfc-1 response

Example GET /collections/{cid}/items response

mfc-1

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": "mf-1",
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [139.757083, 35.627701, 0.5],
          [139.757399, 35.627701, 2.0],
          [139.757555, 35.627688, 4.0],
          [139.757651, 35.627596, 4.0],
          [139.757716, 35.627483, 4.0]
        ]
      },
      "properties": {
        "name": "car1",
        "state": "test1",
        "video": "http://www.opengis.net/spec/movingfeatures/json/1.0/prism/example/video.mpeg"
      },
      "bbox": [
        139.757083, 35.627483, 0.0,
        139.757716, 35.627701, 4.5
      ],
      "time": [
        "2011-07-14T22:01:01Z",
        "2011-07-15T01:11:22Z"
      ],
      "crs": {
        "type": "Name",
        "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"
      },
      "trs": {
        "type": "Name",
        "properties": "urn:ogc:data:time:iso8601"
      }
    }
  ]
}
```

MovingFeature Resource Instance
 → GET /collections/mfc-1/items/mf-1 response
 → Almost the same schema as MovingFeature object in MF-JSON

We can get MovingFeatureCollection for Collection “mfc-1”
 (List of the static information of the moving feature)
 (Does not include TemporalGeometry and TemporalProperties)

```
"crs": {
  "type": "Name",
  "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"
},
"trs": {
  "type": "Name",
  "properties": "urn:ogc:data:time:iso8601"
},
"links": [
  {
    "href": "https://data.example.org/collections/mfc-1/items",
    "rel": "self",
    "type": "application/geo+json"
  },
  {
    "href": "https://data.example.org/collections/mfc-1/items&offset=1&limit=1",
    "rel": "next",
    "type": "application/geo+json"
  }
],
"timeStamp": "2020-01-01T12:00:00Z",
"numberMatched": 100,
"numberReturned": 1
}
```

Example GET /collections/{cid}/items with *subTrajectory* parameter

MC-1

```
"type": "FeatureCollection",
"features": [
  {
    "id": "m",
    "type": "temporalGeometry",
    "type": "MovingPoint",
    "datetimes": ["2011-07-14T22:01:01Z", "2011-07-14T22:01:02Z", "2011-07-14T22:01:03Z",
    "2011-07-14T22:01:04Z", "2011-07-14T22:01:05Z"],
    "coordinates": [
      [139.757083, 35.627701, 0.5],
      [139.757399, 35.627701, 2.0],
      [139.757555, 35.627688, 4.0],
      [139.757651, 35.627596, 4.0],
      [139.757716, 35.627483, 4.0]
    ],
    "interpolation": "Linear"
  },
  {
    "properties": {
      "label": "car",
      "state": "test1",
      "video": "http://www.opengis.net/spec/movingfeatures/json/1.0/prism/example/video.mp4"
    },
    "bbox": [
      139.757083, 35.627483, 0.0,
      139.757716, 35.627701, 4.5
    ],
    "time": [
      "2011-07-14T22:01:01Z",
      "2011-07-15T01:11:22Z"
    ],
    "crs": {
      "type": "Name",
      "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"
    },
    "trs": {
      "type": "Name",
      "properties": "urn:ogc:data:time:iso8601"
    }
  }
]
```

When using the *subTrajectory* query

We can get MovingFeatureCollection with the same schema as *MovingFeature* object in MF-JSON

When using the *subTrajectory* query parameter as "true" with a GET operation, we can get MovingFeatureCollection with TemporalGeometry for Collection "mfc-1"

```
"crs": {
    "type": "Name",
    "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"
},
"trs": {
    "type": "Name",
    "properties": "urn:ogc:data:time:iso8601"
},
"links": [
    {
        "href": "https://data.example.org/collections/mfc-1/items",
        "rel": "self",
        "type": "application/geo+json"
    },
    {
        "href": "https://data.example.org/collections/mfc-1/items&offset=1&limit=1",
        "rel": "next",
        "type": "application/geo+json"
    }
],
"timeStamp": "2020-01-01T12:00:00Z",
"numberMatched": 100,
"numberReturned": 1
```

Example GET /collections/{cid}/items/{mid}/tgsequence

```
{  
  "type": "TemporalGeometrySequence",  
  "geometrySequence": [  
    {  
      "id": "tg-1",  
      "type": "MovingPoint",  
      "datetimes": [  
        "2011-07-14T22:01:02Z",  
        "2011-07-14T22:01:03Z",  
        "2011-07-14T22:01:04Z"  
      ],  
      "coordinates": [  
        [139.757399, 35.627701, 2.0],  
        [139.757555, 35.627688, 4.0],  
        [139.757651, 35.627596, 4.0]  
      ],  
      "interpolation": "Linear",  
      "base": {  
        "type": "glTF",  
        "href": "https://www.opengis.net/spec/movingfeatures/json/1.0/prism/example/car3dmodel.gltf"  
      },  
      "orientations": [  
        {  
          "scales": [1,1,1],  
          "angles": [0,355,0]  
        },  
        {  
          "scales": [1,1,1],  
          "angles": [0,0,330]  
        },  
        {  
          "scales": [1,1,1],  
          "angles": [0,0,300]  
        }  
      ],  
      "crs": {  
        "type": "Name",  
        "properties": "urn:ogc:def:crs:OGC:1.3:CRS84"  
      },  
      "trs": {  
        "type": "Name",  
        "properties": "urn:ogc:def:transform:OGC:1.3:TR84"  
      }  
    }  
  ]  
}
```

mfc-1

mf-1

We can get TemporalGeometrySequence for MovingFeature “mf-1”
(Sequence of the temporal primitive geometry with timestamps)

TemporalPrimitiveGeometry Resource Instance

→ GET /collections/mfc-1/items/mf-1/tgsequence/tg-1 response

→ same schema as the TemporalPrimitiveGeometry object in MF-JSON

```
"links": [  
  {  
    "href": "https://data.example.org/collections/mfc-1/items/mf-1/tgsequence",  
    "rel": "self",  
    "type": "application/json"  
  },  
  {  
    "href": "https://data.example.org/collections/mfc-1/items/mf-1/tgsequence&offset=10&limit=1",  
    "rel": "next",  
    "type": "application/json"  
  }  
,  
  "timeStamp": "2021-09-01T12:00:00Z",  
  "numberMatched": 100,  
  "numberReturned": 1  
]
```

Example GET /collections/{cid}/items/{mid}/tproperties

mfc-1

mf-1

```
{  
  "temporalProperties": [  
    {  
      "name": "length",  
      "type": "TReal",  
      "form": "http://www.qudt.org/qudt/owl/1.0.0/quantity/Length"  
    },  
    {  
      "name": "speed",  
      "type": "TReal",  
      "form": "KHM"  
    }  
  ],  
  "links": [  
    {  
      "href": "https://data.example.org/collections/mfc-1/items/mf-1/tproperties",  
      "rel": "self",  
      "type": "application/json"  
    },  
    {  
      "href": "https://data.example.org/collections/mfc-1/items/mf-1/tproperties&offset=2&limit=2",  
      "rel": "next",  
      "type": "application/json"  
    }  
  ],  
  "timeStamp": "2021-09-01T12:00:00Z",  
  "numberMatched": 10,  
  "numberReturned": 2  
}
```

We can get TemporalProperties for MovingFeature “mf-1”
(List of the metadata of temporal property)
(Doesn’t include temporal property values)
(Not the same as *TemporalProperties* object in MF-JSON)

Example GET /collections/{cid}/items/{mid}/tproperties with **subTemporalValue** query parameter =true

```
{  
  "temporalProperties": [  
    {  
      "datetimes": ["2011-07-14T22:01:06.000Z", "2011-07-14T22:01:07.000Z", "2011-07-  
14T22:01:08.000Z"],  
      "length": {  
        "type": "Measure",  
        "form": "http://www.qudt.org/qudt/owl/1.0.0/quantity/Length",  
        "values": [1.0, 2.4, 1.0],  
        "interpolation": "Linear"  
      },  
      "speed" : {  
        "type" : "Measure",  
        "form" : "KMH",  
        "values" : [65.0, 70.0, 80.0],  
        "interpolation": "Linear"  
      }  
    }  
  ],  
  "links": [  
    {  
      "href": "https://data.example.org/collections/mfc-1/items/mf-1/tproperties",  
      "rel": "self",  
      "type": "application/json"  
    }  
  ],  
  "timeStamp": "2021-09-01T12:00:00Z",  
  "numberMatched": 10,  
  "numberReturned": 2  
}
```

TemporalProperties object in MF-JSON

When using the **subTemporalValue** query parameter as "true" with a GET operation,
we can get TemporalProperties with temporal property values for MovingFeature "mf-1"

Example GET /collections/{cid}/items/{mid}/tproperties/{name}

mfc-1

mf-1

speed

```
{  
  "temporalProperties": [  
    {  
      "datetimes": [  
        "2011-07-14T22:01:02Z",  
        "2011-07-14T22:01:03Z",  
        "2011-07-14T22:01:04Z"  
      ],  
      "values": [  
        65.0,  
        70.0,  
        80.0  
      ],  
      "interpolation": "Linear"  
    },  
    {  
      "datetimes": [  
        "2011-07-15T08:00:00Z",  
        "2011-07-15T08:00:01Z",  
        "2011-07-15T08:00:02Z"  
      ],  
      "values": [  
        0.0,  
        20.0,  
        50.0  
      ],  
      "interpolation": "Linear"  
    }  
  "links": [  
    {  
      "href": "https://data.example.org/collections/mfc-1/items/mf-1/tproperties/speed",  
      "rel": "self",  
      "type": "application/json"  
    }  
  ]  
}
```

We can get a sequence of the temporal primitive value for TemporalProperty “speed”

OGC MF-JSON supported open-sources

- **OGC MF-JSON** supported open-sources
 - For visualization
 - **STINUUM** (<https://github.com/aistairc/mf-cesium>)
 - A space-time visual analysis tool of moving objects with Cesium
 - For spatio-temporal operation
 - **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



OGC MF-JSON supported open-sources

- **MobilityDB** (<https://github.com/MobilityDB/MobilityDB>)

- Support import and export functions with MF-JSON

- Export:

- `asMFJSON(tpoint,options=0,flags=0,maxdecdigits=15) → bytea`

- Import:

- `tgeompointFromMFJSON(text) → tgeompoint`

- `tgeogpointFromMFJSON(text) → tgeogpoint`

- https://mobilitydb.github.io/MobilityDB/master/ch08.html#tpoint_inout

- **MovingPandas** (<https://github.com/anitagraser/movingpandas>)

- Support import function from MF-JSON

- `read_mf_json(json_file_path, traj_id_property=None, traj_id=0):`

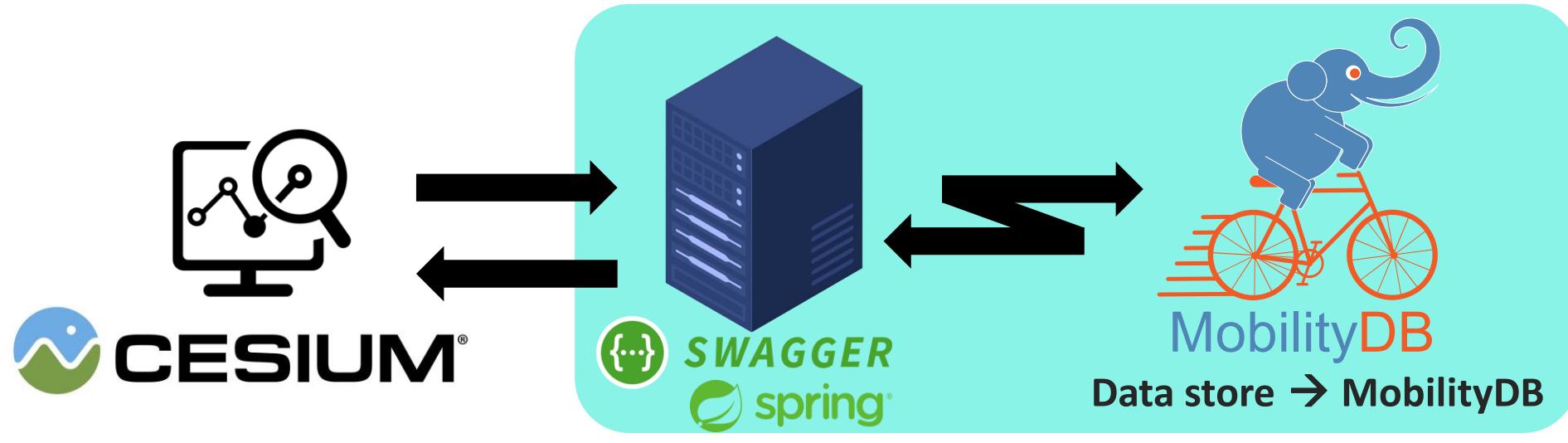
- <https://github.com/movingpandas/movingpandas/blob/main/movingpandas/io.py>



```
150  def read_mf_json(json_file_path, traj_id_property=None, traj_id=0):
151      """
152          Reads OGC Moving Features Encoding Extension JSON files.
153          MovingFeatures files are turned into Trajectory objects.
154          MovingFeatureCollection files are turned into TrajectoryCollection objects.
155          More info: http://www.opengis.net/doc/BP/mf-json/1.0
156
157      Parameters
158      -----
159          json_file_path : str
160              Path to the JSON file
161          traj_id_property : str
162              Name of the MovingFeature JSON property to be used as trajectory ID
163          traj_id : any
164              Trajectory ID value to be used if no traj_id_property is supplied
165
166      Returns
167      -----
168          Trajectory or TrajectoryCollection
169          """
170          with open(json_file_path, "r") as f:
171              data = json.loads(f.read())
172          return read_mf_dict(data, traj_id, traj_id_property)
```

Implementations for OGC API – MF

- Using **OGC MF-JSON** supported open-sources
- **MF-API Server (Opensource): Implement using pygeoapi**
 - Plan to merge into pygeoapi
- . **Use MobilityDB (and PyMEOS) for data storage (in MF-JSON format).**



1. STINUUM with MF-API

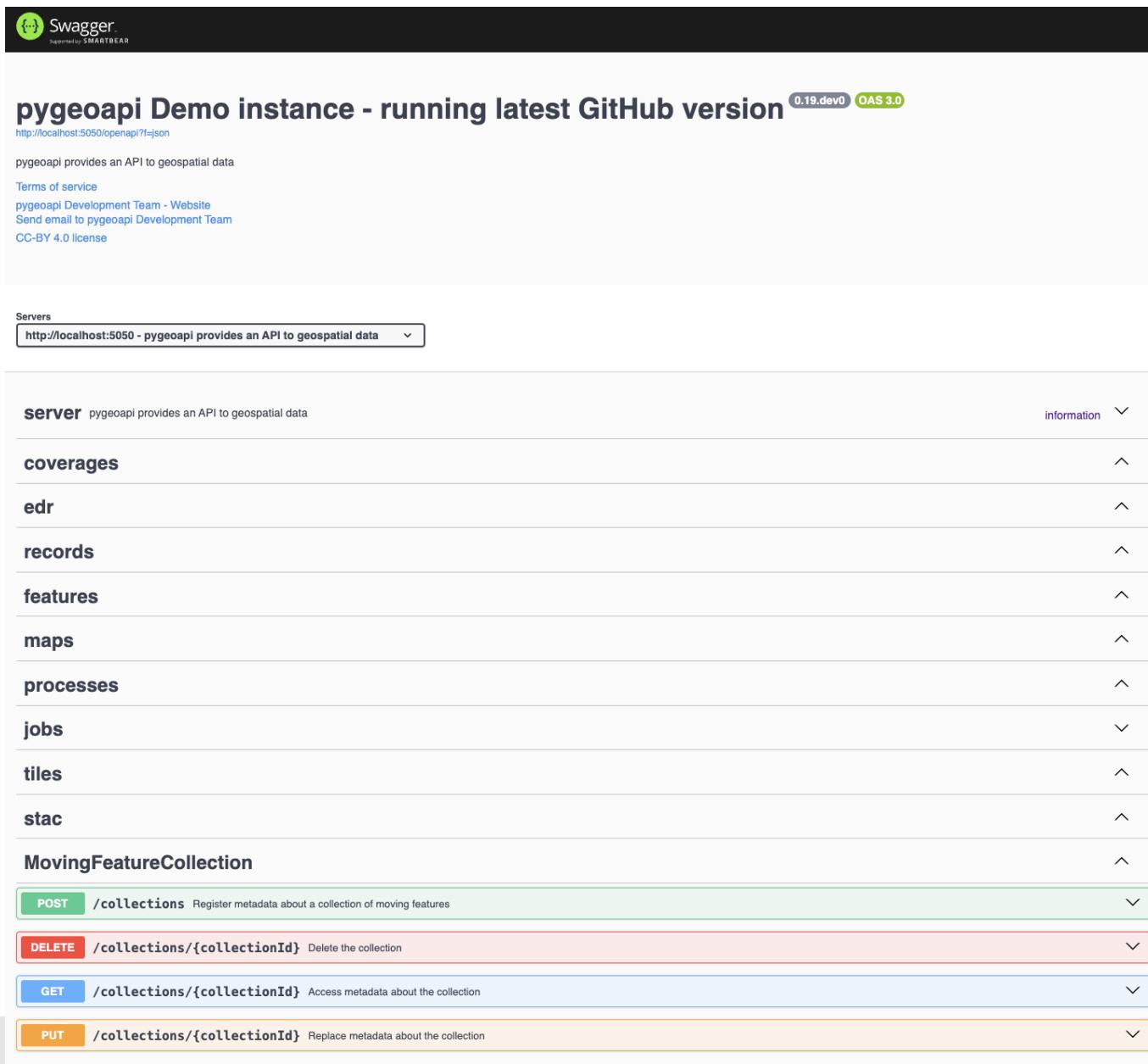


2. MF-API Server (opened as open-source)



MF-API Server (pygeoapi)

- <https://github.com/aistairc/pygeoapi-mf-api>
- Implement mandatory functionality for OGC API – MF
- Easily test OGC API – MF with Swagger
- Let's practice it out in a bit!



The screenshot shows the Swagger UI interface for the pygeoapi Demo instance. At the top, it displays the title "pygeoapi Demo instance - running latest GitHub version 0.19.dev0 OAS 3.0". Below the title, there is a brief description of the API: "pygeoapi provides an API to geospatial data". It includes links for "Terms of service", "pygeoapi Development Team - Website", "Send email to pygeoapi Development Team", and "CC-BY 4.0 license". A "Servers" dropdown is set to "http://localhost:5050 - pygeoapi provides an API to geospatial data". The main content area lists various API endpoints under sections like "server", "coverages", "edr", "records", "features", "maps", "processes", "jobs", "tiles", and "stac". Under the "MovingFeatureCollection" section, four methods are listed: "POST /collections" (green background), "DELETE /collections/{collectionId}" (red background), "GET /collections/{collectionId}" (blue background), and "PUT /collections/{collectionId}" (orange background). Each method has its description and parameters.

OGC API – MF with pygeoapi

All methods are Implemented



Inherited from OGC API-Features
Resources from OGC MF-JSON
Resources from OGC MF-Access

Resource	Path	HTTP Method	Document Reference
Landing page	/	GET	No modification
API definition	/api	GET	
Conformance classes	/conformance	GET	
Collections metadata	/collections	GET, POST	7.3
Collection instance metadata	/collections/{collectionId}	GET, DELETE, PUT	7.4
Moving feature collection	/collections/{collectionId}/items	GET, POST	8.3
Moving feature instance	/collections/{collectionId}/items/{mFeatureId}	GET, DELETE	8.4
Temporal geometry sequence	/collections/{collectionId}/items/{mFeatureId}/tgsequence	GET, POST	8.5
Temporal primitive geometry	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}	DELETE	8.6
Temporal properties	/collections/{collectionId}/items/{mFeatureId}/tproperties	GET, POST	8.8
Temporal property instance	/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}	GET, POST, DELETE	8.9
TemporalPrimitiveValue instance	/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}/{tValueId}	DELETE	8.10
Velocity query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/velocity	GET	8.7
Acceleration query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/acceleration	GET	8.7
Distance query	/collections/{collectionId}/items/{mFeatureId}/tgsequence/{tGeometryId}/distance	GET	8.7

Example: Inserts moving features

A user SHOULD insert a set of moving features or a moving feature into a collection with id `collectionId`.

The request body schema SHALL follows the [MovingFeature object](#) (and [MovingFeatureCollection object](#)) in the OGC MF-JSON.

Parameters

Name	Description
collectionId * required	local identifier of a collection string (path)

Request body

application/json

```
{  
  "type": "Feature",  
  "crs": {  
    "type": "Name",  
    "properties": {"name": "urn:ogc:def:crs:OGC:1.3:CRS84"}  
  },  
  "tr": {  
    "type": "Link",  
    "properties": {  
      "type": "OGCDEF",  
      "href": "http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"  
    }  
  },  
  "temporalGeometry": {  
    "type": "MovingPoint",  
    "datetimes": ["2011-07-14T22:01:01Z", "2011-07-14T22:01:02Z", "2011-07-14T22:01:03Z", "2011-07-14T22:01:04Z"],  
    "coordinates": [[139.757083, 35.627701, 0.5], [139.757399, 35.627701, 2.0], [139.757555, 35.627701, 4.0]],  
    "interpolation": "Linear",  
    "base": {  
      "type": "Time"  
    }  
  }  
}
```

Predefined schema:
MF-JSON MovingFeature object

Execute

Request & Response

Request URL

`http://172.23.145.103:5000/collections/28a4d8df-1a0d-4577-9e06-57b7a0c5a473/items`

Server response

Code Details

201 Response headers

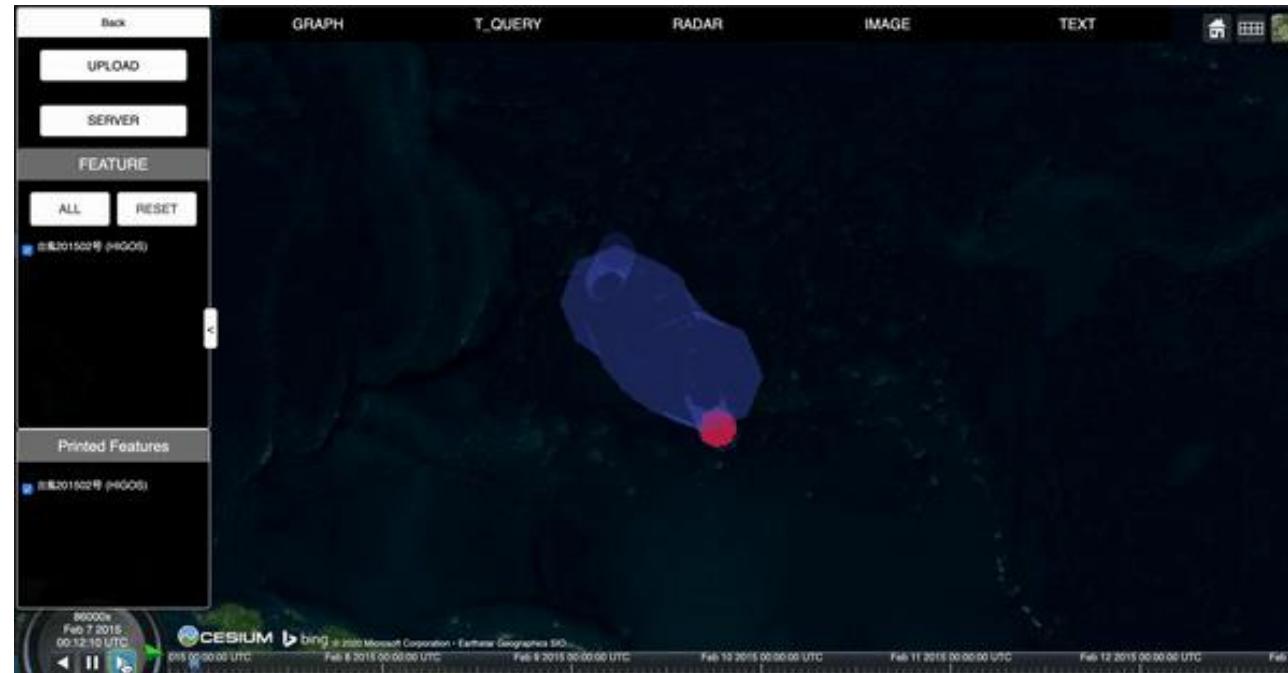
```
access-control-allow-origin: http://172.23.145.103:5000  
connection: close  
content-language: en-US  
content-length: 0  
content-type: application/json  
date: Thu, 26 Jan 2023 04:41:23 GMT  
location: http://172.23.145.103:5000/collections/28a4d8df-1a0d-4577-9e06-57b7a0c5a473/items/5fba5b34-cd3e-45b9-ab27-2a73664c8f3f  
server: Werkzeug/2.2.2 Python/3.7.16  
vary: Origin  
x-powered-by: pygeapi 0.14.dev0
```

Database (with MobilityDB)

Table	Column	Value
mfeature	collection_id	a01a5339-673a-415d-9113-0acddfe95175
mfeature	mfeature_id	ebc5127f-ddcc-433f-a75a-00d24441ea6a
tgeometry	collection_id	a01a5339-673a-415d-9113-0acddfe95175
tgeometry	mfeature_id	ebc5127f-ddcc-433f-a75a-00d24441ea6a
tgeometry	tgeometry_id	c4f4ccc1-ff28-4e1b-b8cf-62c536abda1c
tproperties	collection_id	a01a5339-673a-415d-9113-0acddfe95175
tproperties	mfeature_id	ebc5127f-ddcc-433f-a75a-00d24441ea6a
tproperties	tpropertiesname	255bcb97-c30a-4a27-91ac-4e9d0ab431f3
tproperties	tproperty	[{"length": {"form": "http://www.qudt.org/qudt/owl/1.0.0/quantity"}, "camera": {"type": "Image", "values": [{"http://www.opengis.net/sensorML/1.0.0/quantity": "100m"}]}]

Example: STINUUM with OGC API – MF

- STINUUM supports the visualization of moving features using OGC API – MF (also MF-JSON)
- Let's practice it out soon!



Click the "Server" Button, then move to the data selection page

This screenshot shows the 'GET Server Data' page. It has a table with columns: Number, ID, Type, TemporalGeometry Type, BBox, Time, and Sampling Count. A red box highlights the 'Server' button in the top navigation bar, and an orange arrow points down to the table area. The table displays the message 'No data available in table'.

Get the MF-Collection

This screenshot shows the same 'GET Server Data' page after data has been retrieved. A red box highlights the 'GET Server Data' button in the top navigation bar, and an orange arrow points down to the table area. The table now lists 10 entries, each representing a Feature with a Point geometry. The table includes columns: Number, ID, Type, TemporalGeometry Type, BBox, Time, and Sampling Count. The first entry is numbered 0, ID T0001, Type Feature, and TemporalGeometry Type Point. The table displays the message 'Showing 1 to 10 of 10 entries (Seconds)'.

Result of getting MF-Collection