Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/IS/ogcapi-movingfeatures-1/0.2.1.draft

Internal reference number of this OGC® document: 22-003

Version: 0.2.1.draft

Category: OGC® Implementation Specification

Editor: Taehoon Kim, Kyoung-Sook Kim, Mahmoud SAKR, Martin Desruisseaux

OGC API - Moving Features - Part 1: Features extension

Warning

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

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

Document type:	OGC® Implementation Specification
Document stage:	Draft
Document language:	English

License Agreement

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

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

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

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

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

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

Table of Contents

i. Abstract

Table 1. Overview of Resources

Resource	Path	HTTP Method	Document Reference
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
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

ii. Keywords

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

ogcdoc, OGC document, OGC MovingFeature, MovingFeatures JSON, MovingFeature Access, API, OpenAPI, REST, trajectory

iii. Preface

OGC Declaration

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

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

- Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology
- Université libre de Bruxelles
- Geomatys

v. Submitters

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

Name	Organization
Kyoung-Sook KIM	Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology
Taehoon KIM	Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology
Mahmoud SAKR	Université libre de Bruxelles
Martin Desruisseaux	Geomatys

Chapter 1. Scope

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

Chapter 2. Conformance

This Standard defines XXXX.

Requirements for N standardization target types are considered: * AAAA * BBBB

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

In order to conform to this OGC® Standard, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative). * Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

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

Chapter 3. References

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

Chapter 4. Terms and Definitions

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

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

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

application programming interface (API)

a formally defined set of types and methods which establish a contract between client code which uses the API and implementation code which provides the API

coordinate

one of a sequence of numbers designating the position of a point

Note 1 to entry: In a spatial coordinate reference system, the coordinate numbers are qualified by units.

[source: ISO 19111]

coordinate reference system (CRS)

coordinate system that is related to an object by a datum

Note 1 to entry: Geodetic and vertical datums are referred to as reference frames.

Note 2 to entry: For geodetic and vertical reference frames, the object will be the Earth. In planetary applications, geodetic and vertical reference frames may be applied to other celestial bodies.

[source: ISO 19111]

dataset

identifiable collection of data

[source: ISO 19115-1]

datatype

specification of a value domain with operations allowed on values in this domain

Examples: `Integer`, `Real`, `Boolean`, `String` and `Date`.

Note 1 to entry: Data types include primitive predefined types and user definable types.

[source: ISO 19103]

dynamic attribute

characteristic of a feature in which its value varies with time

[source: OGC 16-140]

feature

abstraction of a real world phenomena

Note 1 to entry: A feature can occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

[source: ISO 19109]

feature attribute

characteristic of a feature

Note 1 to entry: A feature attribute can occur as a type or an instance. Feature attribute type or feature attribute instance is used when only one is meant.

[source: ISO 19109]

feature table

table where the columns represent feature attributes, and the rows represent features

[source: OGC 06-104]

geographic feature

representation of real world phenomenon associated with a location relative to the Earth

[source: ISO 19101-2]

geometric object

spatial object representing a geometric set

[source: ISO 19107:2003]

moving feature

feature whose location changes over time

Note 1 to entry: Its base representation uses a local origin and local coordinate vectors of a geometric object at a given reference time.

Note 2 to entry: The local origin and ordinate vectors establish an engineering coordinate reference system (ISO 19111), also called a local frame or a local Euclidean coordinate system.

property

facet or attribute of an object referenced by a name

[source: ISO 19143]

trajectory

path of a moving point described by a one parameter set of points

[source: ISO 19141]

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this Standard are denoted by the URI

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

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

Chapter 6. Overview

6.1. General

OGC API standards enable access to resources using the HTTP protocol and its associated operations (GET, PUT, POST, DELETE, etc.) OGC API-Common defines a set of features which are applicable to all OGC APIs. Other OGC standards extend API-Common with features specific to a resource type.

This OGC API-Moving Features-Part1:Feature Extension standard defines an API with a goal:

- Provide interface for create, retrieve, update, and delete to ***Moving Features***, which conformance to the OGC Moving Features JSON encoding standard

Resources exposed through an OGC API may be accessed through a Universal Resource Identifier (URI). URIs are composed of three sections:

- Dataset distribution API: The endpoint corresponding to a dataset distribution, where the landing page resource as defined in OGC API-Common-Part 1: Core is available (subsequently referred to as Base URI or `{root}`)
- Access Paths: Unique paths to Resources
- Query Parameters: Parameters to adjust the representation of a Resource or Resources like encoding format or sub-setting

Access Paths are used to build resource identifiers. It is recommended, but not required. Most resources are also accessible through links on previously accessed resources. Unique relation types are used for each resource.

Table 2 summarizes the access paths and relation types defined in this standard.

Table 2. Moving Features API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page for this dataset distribution
{root}/api	`service-desc` or `service-doc`	API Description
{root}/conformance	`conformance`	Conformance Classes
Collections		
{root}/collections	`data`	Metadata describing the Collection Catalog of data available from this API.
{root}/collections/{collectionId}		Metadata describing the Collection Catalog of data which has the unique identifier `{collectionId}`
Moving Features		

Path Template	Relation	Resource
{root}/collections/{collectionId}/items	items	Static information of MovingFeature about available items in the specified Collection
{root}/collections/{collectionId}/items/{mFeatureId}	item	Static information describing the MovingFeature of data which has the unique identifier {mFeatureId}
{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries	items	Temporal object information of TemporalGeometryCollection about available items in the specified MovingFeature
{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries/{tGeometryId}	item	Temporal object describing the TemporalGeometryCollection of data which has the unique identifier {tGeometryId}
{root}/collections/{collectionId}/items/{mFeatureId}/tproperties	items	Temporal object information of TemporalPropertyCollection about available items in the specified MovingFeature
{root}/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}	item	Temporal object describing the TemporalPropertyCollection of data which has the unique identifier {tPropertyName}

Where:

- {root} = Base URI for the API server
- {collectionId} = An identifier for a specific Collection of data
- {mFeatureId} = An identifier for a specific MovingFeature of a specific Collection of data
- {tGeometryId} = An identifier for a specific TemporalGeometry of a specific MovingFeatures of data
- {tPropertyName} = An identifier for a specific TemporalProperty of a specific MovingFeatures of data

Figure 1 shows a UML class diagram for OGC API-MF which represents the basic resources of this standard, such as Collections, Collection, MovingFeature, TemporalGeometry, TemporalGeometryCollection, TemporalPropertyCollection, and TemporalProperty. In this standard, a single moving feature can have temporal geometries, such as a set of trajectories. Also, the moving feature can have temporal properties, such as a set of parametric values.

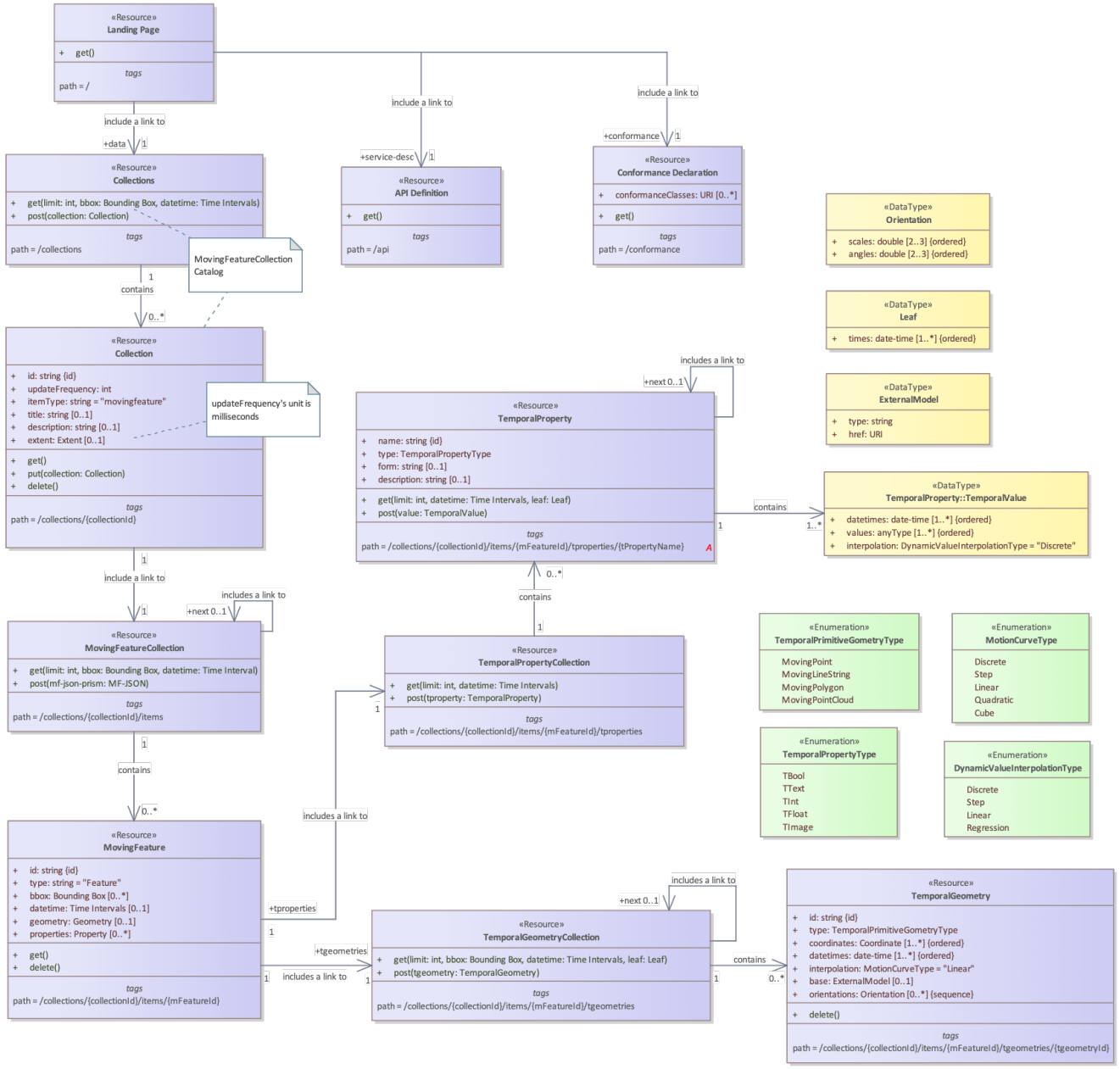

Figure 1. Class diagram for OGC API-MF

6.2. Search

The core search capability is based on OGC API-Common and thus supports:

- bounding box searches,
- time instant or time period searches,
- and equality predicates (i.e. *property=value*).

OGC API-Moving Features extends these core search capabilities to include:

- find leaf value with time instant

6.3. Dependencies

The OGC API-Moving Features (shortly, OGC API-MF) standard is an extension of the OGC API-Common and the OGC API-Features standards. Therefore, an implementation of OGC API-MF shall first satisfy the appropriate Requirements Classes from API-Common and API-Features. Also, OGC API-MF standard is based on the OGC Moving Features Encoding Extension for JSON (shortly, OGC MF-JSON) standards. Therefore, an implementation of OGC API-MF shall satisfy the appropriate Requirements Classes from OGC MF-JSON. Table 3, identifies the OGC API - Common and OGC API - Features Requirements Classes which are applicable to each section of this Standard. Instructions on when and how to apply these Requirements Classes are provided in each section.

Table 3. Mapping OGC API-MF Sections to OGC API - Common, OGC API - Features, and OGC MF-JSON Requirements Classes

API-MF Section	API-MF Requirements Class	API - Common, API - Features, MF-JSON Requirements Class
API Landing Page	http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/common	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/landing-page
API Definition	http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/common	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/landing-page
Declaration of Conformance Classes	http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/common	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/landing-page
Collections	http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/mf-collection	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections, http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete
MovingFeatures	http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/movingfeatures	http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core, http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete, http://www.opengis.net/spec/movingfeatures/json/1.0/req/trajectory, http://www.opengis.net/spec/movingfeatures/json/1.0/req/prism
OpenAPI 3.0		http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30
GeoJSON		http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/geojson

OGC Moving Features JSON		http://www.opengis.net/spec/movingfeatures/json/1.0/req/prism, http://www.opengis.net/spec/movingfeatures/json/1.0/req/prism

Chapter 7. Requirements Class "Common"

7.1. Overview

7.2. API landing page

The landing page provides links to start exploration of the resources offered by an API. OGC API - Common already requires some common links, sufficient for this standard, that are stated in the following Requirements Class of OGC API-Common:

Dependencies

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
- Landing Page, http://www.opengis.net/spec/ogcapi-common/1.0/req/landing-page

7.2.1. Operation

The `Landing Page` operation is defined in the `Landing Page` conformance class of OGC API—Common. No modifications are required. The `Landing Page` conformance class specifies only one way of performing this operation:

1. Issue a `GET` request on the `{root}/` path

Support for `GET` on the `{root}/` path is required by OGC API—Common.

7.2.2. Response

A successful response to the `Landing Page` operation is defined in OGC API-Common. No modifications are required. The schema for this resource is provided in Landing Page Response Schema.

Landing Page Response Schema

```
type: object
required:
  - links
properties:
  title:
    type: string
    example: Movingfeatures data server
  description:
    type: string
    example: Access to data about moving features
  links:
    type: array
    items:
      $ref: ogcapi-features-core/link.yaml
```

The following JSON fragment is an example of a response to an OGC API-MF `Landing Page` operation

Landing Page Example

```
{
  "title": "Movingfeatures data server",
  "description": "Access to data about moving features",
  "links": [
    {
      "href": "http://data.example.com/movingfeatures/mf-123",
      "rel": "alternate",
      "type": "application/geo+json",
      "hreflang": "en",
      "title": "moving features",
      "length": 0
    }
  ]
}
```

7.2.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

7.3. API definition

Every API is required to provide a definition document that describes the capabilities of that API. This definition document can be used by developers to understand the API, by software clients to connect to the server, or by development tools to support the implementation of servers and clients.

Support for an API definition is specified in the following Requirements Class of OGC API - Common:

Dependencies

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
- Landing Page, http://www.opengis.net/spec/ogcapi-common/1.0/req/landing-page

7.3.1. Operation

This operation is defined in the `Landing Page` conformance class of OGC API—Common. No modifications are required. The `Landing Page` conformance class describes two ways of performing this operation:

1. Issue a `GET` request on the `{root}/api` path
2. Follow the `service-desc` or `service-doc` link on the landing page

Only the link is required by OGC API—Common.

7.3.2. Response

A successful response to the API Definition request is a resource which documents the design of the API. OGC API—Common leaves the selection of format for the API Definition response to the API implementor. However, the options are limited to those which have been defined in the OGC API-Common standard. At this time OpenAPI 3.0 is the only option provided.

7.3.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

7.4. Declaration of conformance classes

To support "generic" clients that want to access multiple OGC API standards and extensions - and not "just" a specific API server, the API has to declare the conformance classes it claims to have implemented.

Support for the declaration of conformance classes is specified in the following Requirements Class of OGC API - Common:

Dependencies

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
- Landing Page, http://www.opengis.net/spec/ogcapi-common/1.0/req/landing-page

7.4.1. Operation

This operation is defined in the `Landing Page` conformance class of OGC API—Common. No modifications are required. The `Landing Page` conformance class describes two ways of performing this operation:

1. Issue a `GET` request on the `{root}/conformance` path
2. Follow the `conformance` link on the landing page

Both techniques are required by OGC API—Common.

7.4.2. Response

A successful response to the `Conformance` operation is defined in OGC API-Common. No modifications are required. The schema for this resource is provided in Conformance Response Schema.

Conformance Response Schema

```
type: object
required:
  - conformsTo
properties:
  conformsTo:
    type: array
    items:
      type: string
      example:
        - "http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/conf/common"
        - "http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/conf/mf-collection"
        - "http://www.opengis.net/spec/ogcapi-movingfeatures-
1/1.0/conf/movingfeatures"
```

The following JSON fragment is an example of a response to an OGC API-MF `Conformance` operation

Conformance Example

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/conf/common",
    "http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/conf/mf-collection",
    "http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/conf/movingfeatures"
  ]
}
```

7.4.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

7.5. Parameters

The query parameters `bbox`, `datetime` and `limit` are inherited from API - Common. All requirements and recommendations in API - Common regarding these parameters also apply for API - Moving Features.

7.5.1. Parameter limit

Requirement 1	**/req/common/param-limit**
A	A OGC API-MF SHALL support the Limit parameter for the operation.

B	Requests which include the Limit parameter SHALL comply with API-Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-limit-definition.
C	Responses to Limit requests SHALL comply with API-Common requirements http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-limit-response

7.5.2. Parameter bbox

Requirement 2	**/req/common/param-bbox**
A	A OGC API-MF SHALL support the Bounding Box (bbox) parameter for the operation.
B	Requests which include the Bounding Box parameter SHALL comply with OGC API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-bbox-definition.
C	Responses to Bounding Box requests SHALL comply with OGC API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-bbox-response.

7.5.3. Parameter datetime

Requirement 3	**/req/common/param-datetime**
A	A OGC API-MF SHALL support the DateTime (datetime) parameter for the operation.
B	Requests which include the DateTime parameter SHALL comply with OGC API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-time-definition.
C	Responses to DateTime requests SHALL comply with OGC API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-time-response.

Chapter 8. Requirements Class "Collection Catalog"

8.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/mf-collection	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections
Dependency	http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete

The `Collection Catalog` requirements class defines the requirements for a collection. A collection is an object that provides information about and access to a set of related MovingFeature.

8.2. Information Resources

The two resources defined in this Requirements Class are summarized in Table 4.

Table 4. Collection Catalog Resources

Resource	URI	HTTP Method	Description
Collections	`{root}/collections`	GET	Get information which describes the set of available Collections
		POST	Add a new resource (Collection) instance to a Collections
Collection	`{root}/collections/{collectionId}`	GET	Get information about a specific Collection ({collectinoId}) of geospatial data with links to distribution
		PUT	Update information about a specific Collection ({collectinoId})
		DELETE	Delete a specific Collection ({collectinoId})

8.3. Resource Collections

The `Collections` resource supports retrieving and creating operations via GET and POST HTTP methods respectively.

1. Retrieving operation returns a set of metadata which describes the collections available from this API. The catalog of collections returned to the response can be limited using the `limit`, `bbox`,

and `datetime` parameters.

2. Creating operation post a new Collection resource instance to the collections with this API.

8.3.1. Operations

Retrieve

This operation is defined in the `Collections` conformance class of API-Common. No modifications are needed to support `MovingFeature` resources.

1. Issue a `GET` request on `{root}/collections` path

Support for HTTP GET method on the `{root}/collections` path is required by API-Common.

Requirement 4	/req/mf-collection/collections-op/get
A	The API implementation SHALL comply with the API-Common `Collections` operation requirement `http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`.
B	The API-Common `rec/collections/rc-md-item-type` recommendation SHALL apply as collection's `itemType` property is specified as `movingfeature`.

Create

This operation is defined in the `CREATE` conformance class of API-Features. This operation targeted Collection resource.

1. Issue a `POST` request on `{root}/collections` path

Support for HTTP POST method is required by API-Features.

Requirement 5	/req/mf-collection/collections-op/post
A	The API implementation SHALL comply with the API-Feature `CREATE` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
B	The API implementation SHALL comply with the API-Feature `CREATE` request body requirements `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
C	The content of the request body SHALL be based upon the `Collection` request body schema.

Collection Request Body Schema

```
type: object
required:
  - updateFrequency
properties:
  title:
    description: human readable title of the collection
    type: string
  updateFrequency:
    description: a time interval of sampling location. The unit is millisecond.
    type: number
  description:
    description: any description
    type: string
```

The following example adds a new feature (collection information) to the feature collections. The feature is represented as JSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Create a New Collection Example

```
   Client                                                          Server
     |                                                               |
     | POST /collections   HTTP/1.1                                  |
     | Content-Type: application/json                                |
     |                                                               |
     | {                                                             |
     |   "title": "MovingFeatureCollection_1",                       |
     |   "updateFrequency": 1000,                                    |
     |   "description": "a collection of moving features to manage data  |
     |                   in a distinct (physical or logical) space"  |
     | }                                                             |
     |-------------------------------------------------------------->|
     |                                                               |
     | HTTP/1.1 201 Created                                          |
     | Location: /collections/mfc_1                                  |
     |<--------------------------------------------------------------|
```

8.3.2. Response

Retrieve

A successful response to the `Collections` GET operation is a document that contains summary metadata for each collection accessible through the API. In a typical API deployment, the `Collections` GET response will list collections of all offered resource types. The collections where the value of the `itemType` property is `movingfeature` are collections of moving features.

Requirement 6	/req/mf-collection/collections-response/get
A	The API implementation SHALL comply with the API-Common Collections response requirement http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections.
B	The content of that response SHALL be based upon the Collections response schema.

Collections GET Response Schema

```
type: object
required:
  - collections
  - links
properties:
  collections:
    type: array
    items:
      $ref: collection.yaml
  links:
    type: array
    items:
      $ref: link.yaml
```

The following JSON payload is an example of a response to an OGC API-Moving Features Collections GET operation.

Collections Example


```
{
  "collections": [
    {
      "id": "mfc-1",
      "title": "moving_feature_collection_sample",
      "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"
    }
  ]
}
```


Create

A successful response to the `Collections` POST operation is an HTTP status code.

Requirement 7	/req/mf-collection/collections-response/post
A	The API implementation SHALL comply with the API-Feature `CREATE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

8.3.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

8.4. Resource Collection

A Collection information object is the set of metadata that describes a single collection. An abbreviated copy of this information is returned for each `Collection` in the `{root}/collections` GET response.

The schema for the collection information object presented in this clause is an extension of the collection schema defined in OGC API-Common and OGC API-Features.

Table 5 defines the set of properties that may be used to describe a collection.

Table 5. Table of collection properties

Property	Requirement	Description
id	M	A unique identifier to the collection.
title	O	A human-readable name given to the collection.
description	O	A free-text description of the collection.
links	M	A list of links for navigating the API (e.g. link to previous or next pages; links to alternative representations, etc.)
extent	O	The spatio-temporal coverage of the collection.
itemType	M	Fixed to the value "movingfeature".
updateFrequency	M	A time interval of sampling location. The time unit of this property is second.

NOTE The properties *id*, *title*, *description*, *links*, *extent*, and *itemsType* were inherited from OGC API-Common and OGC API-Features.

NOTE	An update frequency is one of the most important properties of moving feature collection. It is determined by a data source. It can use to determine the continuity of the moving feature's trajectory.

Requirement 8	/req/mf-collection/mandatory-collection
A	A collection object SHALL contain all the mandatory properties listed in Table 5.

8.4.1. Operation

Retrieve

This operation is defined in the `Collection` conformance class of API-Common. No modifications are required to support `MovingFeature` resources.

1. Issue a `GET` request on the `{root}/collections/{collectionId}` path

The {collectionId} parameter is the unique identifier for a single collection offered by the API. The list of valid values for `{collectionId}` is provided in the `/collections` response.

Support for the `{root}/collections/{collectionId}` path is required by OGC API-Common.

Requirement 9	/req/mf-collection/collection-op/get
A	The API implementation SHALL comply with the API-Common `Collection` operation requirement `http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections/src-md-op`.
B	The API-Common `/rec/collections/rc-md-item-type` recommendation SHALL apply to collections where the value of the `itemType` property is specified as `movingfeature`.

Replace

This operation is defined in the `REPLACE` conformance class of API-Features. This operation targeted Collection resource.

1. Issue a `PUT` request on `{root}/collections/{collectionId}` path

Support for HTTP PUT method is required by API-Features.

Requirement 10	/req/mf-collection/collection-op/put
A	The API implementation SHALL comply with the API-Feature `PUT` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/update-put-put-op`.

Requirement 10	/req/mf-collection/collection-op/put
B	The API implementation SHALL comply with the API-Feature PUT request body requirements http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/update-put-*.
C	The content of the request body SHALL be based upon the Collection request body schema, except updateFrequency. If the updateFrequency is included in the request body, the server SHALL ignore it.

NOTE The update frequency cannot be changed once set.

The following example replaces the feature created by the Create Example with a new feature (collection information without an update frequency). Once again, the replacement feature is represented as JSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Replace an Existing Collection Example

Delete

This operation is defined in the DELETE conformance class of API-Features.

1. Issue a DELETE request on {root}/collections/{collectionId} path

Support for HTTP DELETE method is required by API-Features.

Requirement 11	/req/mf-collection/collection-op/delete
A	The API implementation SHALL comply with the API-Feature DELETE operation requirement http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete/delete-op.

The following example deletes the feature created by the Create Example and replaced with a new feature in the Replace Example. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Delete an Existing Collection Example

8.4.2. Response

Retrieve

A successful response to the `Collection` GET operation is a set of metadata that describes the collection identified by the `{collectionId}` parameter.

Requirement 12	/req/mf-collection/collection-response/get
A	The API implementation SHALL comply with the API-Common `Collection` response requirement `http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`.
B	The response SHALL only include collection metadata selected by the request.
C	The content of that response SHALL be based upon the `Collection` response schema.

Collection GET Response Schema

```yaml
type: object
required:
  - id
  - links
  - itemType
  - updateFrequency
properties:
  id:
    description: identifier of the collection used, for example, in URIs
    type: string
    example: address
  title:
    description: human readable title of the collection
    type: string
    example: address
  description:
    description: a description of the features in the collection
    type: string
    example: An address.
  links:
    type: array
    items:
      $ref: ogcapi-features-core/link.yaml
    example:
      - href: https://data.example.com/buildings
        rel: item
      - href: https://example.com/concepts/buildings.html
        rel: describedby
        type: text/html
  extent:
    $ref: ogcapi-features-core/extent.yaml
  itemType:
    description: indicator about the type of the items in the collection
    type: string
    default: movingfeature
  crs:
    description: the list of coordinate reference systems supported by the service
    type: array
    items:
      type: string
    default:
      - https://www.opengis.net/def/crs/OGC/1.3/CRS84
    example:
      - https://www.opengis.net/def/crs/OGC/1.3/CRS84
      - https://www.opengis.net/def/crs/EPSG/0/4326
  updateFrequency:
    description: a time interval of sampling location
    type: number
```

The following JSON payload is an example of a response to an OGC API-Moving Features `Collection` GET operation.

Collection Information Example


```
{
  "id": "mfc-1",
  "title": "moving_feature_collection_sample",
  "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"
    }
  ]
}
```


Replace

A successful response to the `Collection` PUT operation is an HTTP status code.

Requirement 13	/req/mf-collection/collection-response/put
A	The API implementation SHALL comply with the API-Feature `PUT` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/update-put-response`.

Requirement 13	/req/mf-collection/collection-response/put
B	The API implementation SHALL comply with the API-Feature `PUT` exception requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/update-put-rid-exception`.

Delete

A successful response to the `Collection` DELETE operation is an HTTP status code.

Requirement 14	/req/mf-collection/collection-response/delete
A	The API implementation SHALL comply with the API-Feature `DELETE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete/response`.
B	If no resource with the identifier exists in the collection, the server SHALL respond with a not-found exception (`404`).

8.4.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

Chapter 9. Requirements Class "Moving Features"

9.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-movingfeatures-1/1.0/req/movingfeatures	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete
Dependency	http://www.opengis.net/spec/movingfeatures/json/1.0/req/trajectory
Dependency	http://www.opengis.net/spec/movingfeatures/json/1.0/req/prism

The `Moving Features` requirements class defines the requirements for a moving feature. A moving feature is an object that provide information about and access to a TemporalGeometryCollection and TemporalPropertyCollection.

9.2. Information Resources

The five resources defined in this Requirements Class are summarized in Table 6.

Table 6. Moving Features Resources

Resource	URI	HTTP Method
MovingFeatures	`{root}/collections/{collectionId}/items`	GET, POST
MovingFeature	`{root}/collections/{collectionId}/items/{mfeatureId}`	GET, DELETE
TemporalGeometryCollection	`{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries`	GET, POST
TemporalGeometry	`{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries/{tGeometryId}`	DELETE
TemporalPropertyCollection	`{root}/collections/{collectionId}/items/{mFeatureId}/tproperties`	GET, POST
TemporalProperty	`{root}/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertiesName}`	GET, POST

9.3. Resource MovingFeatures

The `MovingFeatures` resource supports retrieving and creating operations via GET and POST HTTP methods respectively.

1. Retrieving operation returns a set of features which describes the moving feature available from this API.
2. Creating operation post a new MovingFeature resource instance to a specific Collection (specified by {collectinoId} with this API.

The OGC API-MF `Items` query is an OGC API-Features endpoint that may be used to catalog pre-existing moving features. If a `mFeatureID` is not specified, the query will return a list of the available moving features. The list of moving features returned to the response can be limited using the bbox, datetime, and limit parameters. This behavior is specified in OGC API-Features. All parameters for use with the `Items` query are defined by OGC API-Features.

9.3.1. Operation

Retrieve

This operation is defined in the `MovingFeatures` conformance class of API-Features. No modifications are needed to support MovingFeature resources.

1. Issue a `GET` request on `{root}/collections/{collledctionID}/items` path

Support for GET on the `{root}/collections/{collledctionID}/items` path is required by API-Features.

Requirement 15	/req/movingfeatures/features-op/get
A	The API implementation SHALL comply with the API-Features `Features` operation requirement `http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core/fc-op`.

Create

This operation is defined in the `CREATE` conformance class of API-Features. This operation targeted MovingFeature resource.

1. Issue a `POST` request on `{root}/collections/{collledctionID}/items` path

Support for HTTP POST method is required by API-Features.

Requirement 16	/req/movingfeatures/features-op/post
A	The API implementation SHALL comply with the API-Feature CREATE operation requirement http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete.
B	The API implementation SHALL comply with the API-Feature CREATE request body requirements http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete.
C	The content of the request body SHALL be based upon the MovingFeature object and MovingFeatureCollection object in OGC Moving Features JSON encoding standard schema.

The following example adds a new feature (MovingFeature object in MF-JSON) to the specific Collection. The feature is represented as <OGC-MF-JSON,MF-JSON>>, which is a kind of extension of the GeoJSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Create a New MovingFeature Object Example

```
Client                                                                  Server
  |                                                                       |
  | POST /collections/mfc-1/items   HTTP/1.1                              |
  | Content-Type: application/geo+json                                    |
  |                                                                       |
  | {                                                                     |
  |   "type": "Feature",                                                  |
  |   "id": "mf-1",                                                 |     |
  |   "properties": {                                                     |
  |      "name": "car1",                                                  |
  |      "state": "test1",                                                |
  |      "video": "http://.../example/video.mpeg"                         |
  |   },                                                                  |
  |   "crs": {                                                            |
  |      "type": "Name",                                                  |
  |      "properties": {                                                  |
  |         "name": "urn:ogc:def:crs:OGC:1.3:CRS84"                       |
  |      }                                                                |
  |   },                                                                  |
  |   "trs": {                                                            |
  |      "type": "Link",                                                  |
  |      "properties": {                                                  |
  |         "type": "ogcdef",                                             |
  |         "href": "http://www.opengis.net/def/uom/ISO-8601/0/Gregorian" |
  |      }                                                                |
  |   },                                                                  |
  |   "temporalGeometry": {                                               |
```

```
      "type": "MovingPoint",
      "datetimes": [
         "2011-07-14T22:01:01.000Z",
         "2011-07-14T22:01:02.000Z",
         "2011-07-14T22:01:03.000Z",
         "2011-07-14T22:01:04.000Z",
         "2011-07-14T22:01:05.000Z"
      ],
      "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",
      "base": {
         "type": "glTF",
         "href": "http://.../example/car3dmodel.gltf"
      },
      "orientations": [
          {"scales": [1,1,1],"angles": [0,0,0]},
          {"scales": [1,1,1],"angles": [0,355,0]},
          {"scales": [1,1,1],"angles": [0,0,330]},
          {"scales": [1,1,1],"angles": [0,0,300]},
          {"scales": [1,1,1],"angles": [0,0,270]},
      ]
   },
   "temporalProperties": [
      {
         "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",
            "values": [1,2.4,1],
            "interpolation": "Linear",
            "description": "description1"
         },
         "discharge": {
            "type": "Measure",
            "form": "MQS",
            "values": [3,4,5],
            "interpolation": "Step"
         }
      },
      {
         "datetimes": [
```

```
|            "2011-07-15T23:01:01.450Z",                                  |
|            "2011-07-16T00:01:01.450Z"                                   |
|         ],                                                              |
|         "camera": {                                                     |
|            "type": "Image",                                             |
|            "values": [                                                  |
|               "http://.../example/image1",                              |
|               "VBORw0KGgoAAAANSUhEU......"                              |
|            ],                                                           |
|            "interpolation": "Discrete"                                  |
|         },                                                              |
|         "labels": {                                                     |
|            "type": "Text",                                              |
|            "values": ["car","human"],                                   |
|            "interpolation": "Discrete"                                  |
|         }                                                               |
|      }                                                                  |
|   ]                                                                     |
| }                                                                       |
|------------------------------------------------------------------------>|
|                                                                         |
| HTTP/1.1 201 Created                                                    |
| Location: /collections/mfc-1/items/mf-1                                 |
|<------------------------------------------------------------------------|
```

9.3.2. Response

Retrieve

A successful response to the `MovingFeatures` GET operation is a document that contains the static data of moving features. In a typical API deployment, the `MovingFeatures` GET response will list features of all offered resource types.

Requirement 17	/req/movingfeatures/features-response/get
A	The API implementation SHALL comply with the API-Features `Features` response requirement `http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core/fc-response`.
B	The response SHALL only include moving features selected by the request with parameters.
C	Each moving feature in the response SHALL include the mandatory properties listed in Table 7.

Features GET Response Schema

```yaml
type: object
required:
  - type
  - features
properties:
  type:
    type: string
    enum:
      - FeatureCollection
  features:
    type: array
    items:
      $ref: movingFeatureGeoJSON.yaml
  links:
    type: array
    items:
      $ref: ogcapi-features-core/link.yaml
  timeStamp:
    type: string
    format: date-time
  numberMatched:
    type: integer
    minimum: 0
  numberReturned:
    type: integer
    minimum: 0
```

The following JSON payload is an example of a response to an OGC API-Moving Features `MovingFeatures` GET operation.

MovingFeatures GET Example


```
{
  "type": "FeatureCollection",
  "features":[
    {
      "id": "mf-1",
      "type": "Feature",
      "geometry":{
        "type": "Polygon",
        "coordinates":[
          [-122.308150179, 37.488035566],
          [-122.597502109, 37.538869539],
          [-122.576687533, 37.613537207],
          [-122.2880486, 37.562818007],
          [-122.308150179, 37.488035566]
        ]
      },
      "properties":{
        "label": "car"
      },
      "bbox":[
        -122.59750209, 37.48803556, -122.2880486, 37.613537207
      ],
      "interval":[
        "2011-07-14T22:01:01Z",
        "2011-07-15T01:11:22Z"
      ]
    }
  ],
  "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
}
```


Create

A successful response to the `MovingFeatures` POST operation is an HTTP status code.

Requirement 18	/req/movingfeatures/features-response/post
A	The API implementation SHALL comply with the API-Feature `CREATE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

9.3.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

9.4. Resource MovingFeature

9.4.1. Overview

A MovingFeature object consists of the set of static information that describes a single moving feature and the set of temporal object information, such as temporal geometry and temporal property. An abbreviated copy of this information is returned for each `MovingFeature` in the `{root}/collections/{collectionId}/items` GET response.

The schema for the moving feature object presented in this clause is an extension of the `GeoJSON Feature Object` defined in GeoJSON. Table 7 defines the set of properties that may be used to describe a moving feature.

Table 7. Table of the properties related to the moving feature

Property	Requirement	Description
id	M	A unique record identifier assigned by the server.
type	M	A feature type of GeoJSON (i.e., one of 'Feature' or 'FeatureCollection').
geometry	M	A projective geometry of the moving feature.
properties	O	A set of property of GeoJSON.
bbox	O	A bounding box information for the moving feature.
interval	O	A life span information for the moving feature.
temporalGeometries	O	A set of temporal geometry of the moving feature.
temporalPropertiesCollection	O	A set of temporalProperty of the moving feature.

NOTE	The properties *id*, *type*, *geometry*, *properties*, and *bbox* were inherited from GeoJSON.

Requirement 19	/req/movingfeatures/mandatory-mf
A	A moving feature object SHALL contain all the mandatory properties listed in Table 7.

9.4.2. Operation

Retrieve

This operation is defined in the `Feature` conformance class of API-Features. No modifications are needed to support MovingFeature resources.

1. Issue a `GET` request on the `{root}/collections/{collectionId}/items/{mFeatureId}` path

The {mFeatureId} parameter is the unique identifier for a single moving feature offered by the API. The list of valid values for `{mFeatureId}` is provided in the `{root}/collections/{collectionId}/items` GET response.

Support for GET on the `{root}/collections/{collledctionID}/items/{mFeatureId}` path is required by API-Features.

Requirement 20	/req/movingfeatures/mf-op/get
A	The API implementation SHALL comply with the API-Features `Feature` operation requirement `http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core/f-op`.
B	For every moving feature in a moving feature collection (path `{root}/collections/{collectionId}`), the server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/items/{mFeatureId}`
C	The path parameter `collectionId` is each `id` property in the `Collection` GET operation response where the value of the `itemType` property is specified as `movingfeature`. The path parameter `mFeatureId` is an `id` property of the moving feature.

Delete

This operation is defined in the `DELETE` conformance class of API-Features.

1. Issue a `DELETE` request on `{root}/collections/{collectionId}/items/{mFeatureId}` path

Support for HTTP DELETE method is required by API-Features.

Requirement 21	/req/movingfeatures/mf-op/delete
A	The API implementation SHALL comply with the API-Feature `DELETE` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete-op`.
B	For every moving feature in a moving feature collection (path `{root}/collections/{collectionId}`), the server SHALL support the HTTP DELETE operation at the path `{root}/collections/{collectionId}/items/{mFeatureId}`
C	The path parameter `collectionId` is each `id` property in the `Collection` GET operation response where the value of the `itemType` property is specified as `movingfeature`. The path parameter `mFeatureId` is an `id` property of the moving feature.

9.4.3. Response

Retrieve

A successful response to the `MovingFeature` GET operation is a set of metadata that describes the moving feature identified by the `{mFeatureId}` parameter. This response doesn't include a set of temporal object information. The temporal object information may access by TemporalGeometries and TemporalPropertiesCollection operation.

Requirement 22	/req/movingfeatures/mf-response/get
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The content of that response SHALL include the set of moving feature's metadata that defined in the response schema.

MovingFeature GET Response Schema

```yaml
type: object
required:
  - id
  - type
  - geometry
  - properties
properties:
  id:
    type: string
  type:
    type: string
    enum:
      - Feature
  geometry:
    $ref: ogcapi-features-core/geometryGeoJSON.yaml
  properties:
    type: object
    nullable: true
  bbox:
    type: array
    minItems: 1
    items:
      type: array
      oneOf:
        - minItems: 4
          maxItems: 4
        - minItems: 6
          maxItems: 6
      items:
        type: number
  interval:
    type: array
    minItems: 1
    items:
      type: array
      minItems: 2
      maxItems: 2
      items:
        type: string
        format: date-time
        nullable: true
  links:
    type: array
    items:
      $ref: ogcapi-features-core/link.yaml
```

The `interval` property of the `MovingFeature` response represents a particular period of moving feature existence.

The following JSON payload is an example of a response to an OGC API-MovingFeatures `MovingFeature` operation.

MovingFeature Example


```
{
  "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
  ],
  "interval":[
    "2011-07-14T22:01:01Z",
    "2011-07-15T01:11:22Z"
  ]
}
```


Delete

A successful response to the `Collection` DELETE operation is an HTTP status code.

Requirement 23	/req/movingfeatures/mf-response/delete
A	The API implementation SHALL comply with the API-Feature `DELETE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete/response`.
B	If no resource with the identifier exists in the collection, the server SHALL respond with a not-found exception (`404`).

9.4.4. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

9.5. Resource TemporalGeometryCollection

The `TemporalGeometryCollection` resource supports retrieving and creating operations via GET and POST HTTP methods respectively.

1. Retrieving operation returns a set of temporal geometry object which is included in the `MovingFeature` that specified by {mFeatureId}. The set of temporal geometry object returned to the response can be limited using the `limit`, `bbox`, `datetime`, and `leaf` parameters.
2. Creating operation post a new TemporalGeometry resource to the `MovingFeature` that specified by {mFeatureId}.

9.5.1. Parameters

Parameter leaf

The `leaf` parameter is a sequence of monotonic increasing instants with date-time strings (ex. "2018-02-12T23:20:50Z") that adheres to RFC3339. It consists of a list of the date-time format string, different from `datetime` parameter. The array does not allow the same element.

Example 1. Leaf valid (and invalid) Examples

(O) "2018-02-12T23:20:50Z"

(O) "2018-02-12T23:20:50Z", "2018-02-12T23:30:50Z"

(O) "2018-02-12T23:20:50Z", "2018-02-12T23:30:50Z", "2018-02-12T23:40:50Z"

(X) "2018-02-12T23:20:50Z", "2018-02-12T23:20:50Z"

(X) "2018-02-12T23:20:50Z", "2018-02-12T22:20:50Z"

If `leaf` parameter is provided by the client, the endpoint returns only geometry coordinate (or temporal property value) with the leaf query at each time included in the `leaf` parameter, similar to ***pointAtTime*** operation in the OGC Moving Feature Access standard. And `interpolation` property in the response SHALL be 'Discrete'.

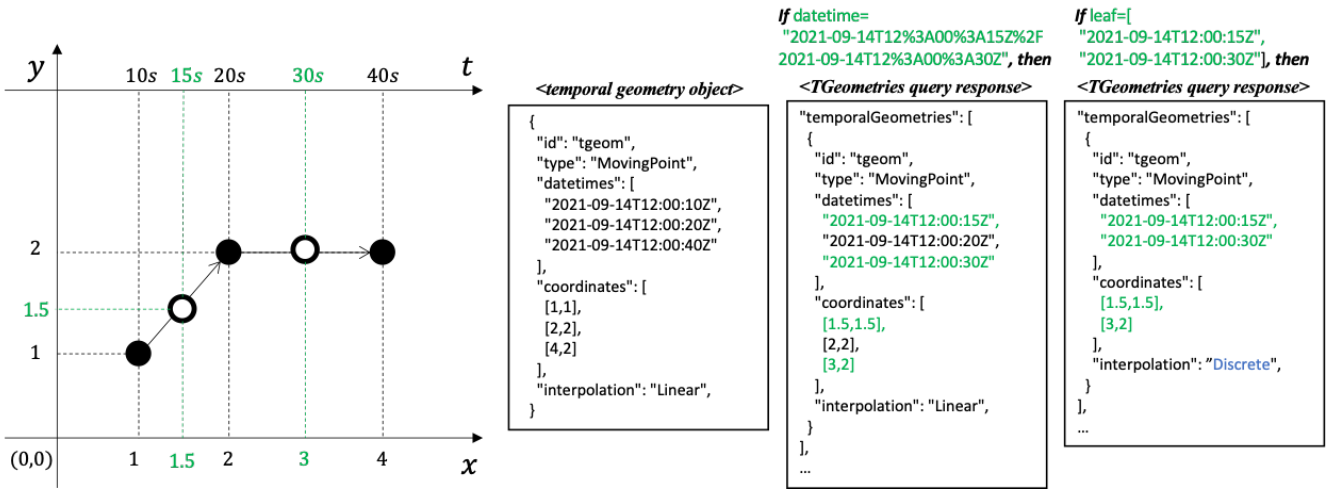

Figure 2. Example of response result with `leaf` parameter

Requirement 24	**/req/movingfeatures/param-leaf-definition**

A	The operation SHALL support a parameter `leaf` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: leaf`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  uniqueItems: true,`
	`  minItems: 1`
	`  items:`
	`  type: string`
	`  format: date-time`
	`style: form`
	`explode: false`

B	The `leaf` parameter SHALL be a sequence of monotonic increasing instants with `date-time` strings.
C	The syntax of `date-time` is specified by RFC 3339, 5.6.

Requirement 25	**/req/movingfeatures/param-leaf-response**
A	If the `leaf` parameter is provided by the client and supported by the server, then only resources that have a temporal information (i.e., `datetimes` property) that intersects the temporal information in the `leaf` parameter SHALL be part of the result set.

Requirement 25	/req/movingfeatures/param-leaf-response
B	The `leaf` parameter SHALL match all resources in the moving feature that are associated with temporal information.
C	If `leaf` parameter is provided by the client and supported by the server, the endpoint SHALL return only temporal geometry coordinate (or temporal property value) with the ***PointAtTime*** query at each time included in the `leaf` parameter, using interpolated trajectory according to the `interpolation` property.
D	If `leaf` parameter is provided by the client and supported by the server, the `interpolation` property in the response SHALL be 'Discrete'.

9.5.2. Operation

Retrieve

1. Issue a `GET` request on the `{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries` path

Requirement 26	/req/movingfeatures/tgeometries-op/get
A	For every moving feature identified in the MovingFeatures GET response (path `{root}/collections/{collectionId}/items`), the server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries`
B	The path parameter `collectionId` is each `id` property in the `Collection` GET response where the value of the `itemType` property is specified as `movingfeature`. The path parameter `mFeatureId` is each `id` property in the `MovingFeatures` GET response.

Create

This operation is defined in the `CREATE` conformance class of API-Features. This operation targeted TemporalGeometry resource.

1. Issue a `POST` request on `{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries` path

Support for HTTP POST method is required by API-Features.

Requirement 27	/req/movingfeatures/tgeometries-op/post
A	The API implementation SHALL comply with the API-Feature `CREATE` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
B	The API implementation SHALL comply with the API-Feature `CREATE` request body requirements `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
C	The content of the request body SHALL be based upon the TemporalGeometry object in OGC Moving Features JSON encoding standard schema.
D	The latest date-time instance in the temporal geometry object in MovingFeature, determined by `mFeatureId`, SHALL be faster than the beginning date-time instance in the temporal geometry object in the request body.

The following example adds a new feature (TemporalGeometry object in MF-JSON) to the feature created by the Creation a MovingFeature Example. The feature is represented as <OGC-MF-JSON,MF-JSON>>, which is a kind of extension of the GeoJSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Create a New TemporalGeometry Object Example

```
Client                                                                  Server
  |                                                                       |
  | POST /collections/mfc-1/items/mf-1/tgeometries   HTTP/1.1             |
  | Content-Type: application/geo+json                                    |
  |                                                                       |
  | {                                                                     |
  |    "id": "tg-1",                                                      |
  |    "type": "MovingPoint",                                             |
  |    "datetimes": [                                                     |
  |       "2011-07-14T22:01:06.000Z",                                     |
  |       "2011-07-14T22:01:07.000Z",                                     |
  |       "2011-07-14T22:01:08.000Z",                                     |
  |    ],                                                                 |
  |    "coordinates": [                                                   |
  |       [139.757716,35.627483,4.0],                                     |
  |       [139.757782,35.627483,4.0],                                     |
  |       [139.757843,35.627483,4.0]                                      |
  |    ],                                                                 |
  |    "interpolation": "Linear",                                         |
  |      "base": {                                                        |
  |         "type": "glTF",                                               |
  |         "href": "http://.../example/car3dmodel.gltf"                  |
  |      },                                                               |
  |      "orientations": [                                                |
  |          {"scales": [1,1,1],"angles": [0,0,270]},                     |
  |          {"scales": [1,1,1],"angles": [0,0,270]},                     |
  |          {"scales": [1,1,1],"angles": [0,0,270]},                     |
  |      ]                                                                |
  |   }                                                                   |
  | }                                                                     |
  |---------------------------------------------------------------------->|
  |                                                                       |
  | HTTP/1.1 201 Created                                                  |
  | Location: /collections/mfc-1/items/mf-1/tgeometries/tg-1              |
  |<----------------------------------------------------------------------|
```

9.5.3. Response

Retrieve

A successful response to the `TemporalGeometryCollection` GET operation is a document that contains the set of temporal geometry of the moving feature identified by the `{mFeatureId}` parameter.

Requirement 28	/req/movingfeatures/tgeometries-response/get
A	The API implementation SHALL comply with the API-Features Features response requirement http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core/fc-response.
B	The response SHALL only include temporal geometries selected by the request with limit, bbox, datetime, and leaf parameters.
C	Each temporal geometry in the response SHALL include the mandatory properties listed in Table 8.

TemporalGeometries GET Response Schema

```
type: object
required:
  - temporalGeometries
properties:
  temporalGeometries:
    type: array
    items:
      $ref: temporalGeometry.yaml
  links:
    type: array
    items:
      $ref: ogcapi-features-core/link.yaml
  timeStamp:
    type: string
    format: date-time
  numberMatched:
    type: integer
    minimum: 0
  numberReturned:
    type: integer
    minimum: 0
```

TemporalGeometry Schema (temporalGeometry.yaml)

```
type: object
required:
  - id
  - type
  - coordinates
  - datetimes
  - interpolation
properties:
  id:
    type: string
```

```yaml
  type:
    type: string
    enum:
      - MovingPoint
      - MovingLineString
      - MovingPolygon
      - MovingPointCloud
  coordinates:
    type: array
    minItems: 2
    items:
      oneOf:
        - $ref: pointCoordinates.yaml
        - $ref: lineStringCoordinates.yaml
        - $ref: polygonCoordinates.yaml
        - $ref: multiPointCoordinates.yaml
  datetimes:
    type: array
    uniqueItems: true,
    minItems: 2
    items:
      type: string
      format: date-time
  interpolation:
    type: string
    enum:
      - Discrete
      - Step
      - Linear
      - Quadratic
      - Cube
  base:
    type: object
    required:
      - href
      - type
    properties:
      href:
        type: string
        format: uri
      type:
        type: string
  orientations:
    type: array
    minItems: 2
    items:
      type: object
      required:
        - scales
        - angles
      properties:
```

```
        scales:
          type: array
          oneOf:
            - minItems: 2
              maxItems: 2
            - minItems: 3
              maxItems: 3
          items:
            type: number
        angles:
          type: array
          oneOf:
            - minItems: 2
              maxItems: 2
            - minItems: 3
              maxItems: 3
          items:
            type: number
```

The following JSON payload is an example of a response to an OGC API-Moving Features `TemporalGeometryCollection` GET operation.

TemporalGeometryCollection GET Example

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


Create

A successful response to the `TemporalGeometryCollection` POST operation is an HTTP status code.

Requirement 29	/req/movingfeatures/tgeometries-response/post
A	The API implementation SHALL comply with the API-Feature `CREATE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

9.5.4. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

9.6. Resource TemporalGeometry

A temporal geometry object represents the movement of a moving feature with various types of moving geometry, i.e., `MovingPoint`, `MovingLineString`, `MovingPolygon`, and `MovingPointCloud`. It can also represent the movement of a 3D object with its orientation.

The schema for the temporal geometry object presented in this clause is an extension of the `TemporalGeometry Object` defined in MF-JSON standard. Table 8 defines the set of properties that may be used to describe a temporal geometry.

Table 8. Table of the properties related to the temporal geometry

Property	Requirement	Description
id	O	An identifier for the resource assigned by an external entity.
type	M	A primitive geometry type of MF-JSON (i.e., one of 'MovingPoint', 'MovingLineString', 'MovingPolygon', 'MovingPointCloud', or 'MovingGeometryCollection').
datetimes	M	A sequence of monotonic increasing instants.
coordinates	M	A sequence of leaf geometries of a temporal geometry, having the same number of elements as "datetimes".
interpolation	M	A predefined type of motion curve (i.e., one of 'Discrete', 'Step', 'Linear', 'Quadratic' or 'Cubic').
base.type	O	A type of 3D file format, such as STL, OBJ, PLY, and glTF.
base.href	O	A URL to address a 3D model data which represents a base geometry of a 3D shape.
orientations.scales	O	An array value of numbers along the x, y, and z axis in order as three scale factors.
orientations.angles	O	An array value of numbers along the x, y, and z axis in order as Euler angles in degree.

NOTE The detailed information and requirements for each property are described in the OGC Moving Feature JSON encoding standard.

Requirement 30	/req/movingfeatures/mandatory-tgeometry
A	A temporal geometry object SHALL contain all the mandatory properties listed in Table 8.

9.6.1. Operation

Delete

This operation is defined in the `DELETE` conformance class of API-Features.

1. Issue a `DELETE` request on `{root}/collections/{collectionId}/items/{mFeatureId}/tgeometries/{tGeometryId}` path

The {tGeometryId} parameter is the unique identifier for a single temporal geometry offered by the API. The list of valid values for {tGeometryId} is provided in the {root}/collections/{collectionId}/items/{mFeatureId}/tgeometries GET response.

Support for HTTP DELETE method is required by API-Features.

Requirement 31	/req/movingfeatures/mf-op/delete
A	The API implementation SHALL comply with the API-Feature DELETE operation requirement http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete/delete-op.
B	For every temporal geometry in a moving feature (path {root}/collections/{collectionId}/items/{mFeatureId}), the server SHALL support the HTTP DELETE operation at the path {root}/collections/{collectionId}/items/{mFeatureId}/tgeometries/{tGeometryId}
C	The path parameter collectionId is each id property in the Collection GET operation response where the value of the itemType property is specified as movingfeature. The path parameter mFeatureId is an id property of the moving feature. The path parameter tGeometryId is an id property of the temporal geometry.

9.6.2. Response

Delete

A successful response to the TemporalGeometry DELETE operation is an HTTP status code.

Requirement 32	/req/movingfeatures/mf-response/delete
A	The API implementation SHALL comply with the API-Feature DELETE response requirement http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete/delete/response.
B	If no resource with the identifier exists in the collection, the server SHALL respond with a not-found exception (404).

9.6.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

9.7. Resource TemporalPropertyCollection

A `TemporalPropertyCollection` object consists of the set of TemporalProperty which is included in the `MovingFeature` that specified by {mFeatureId}. The `TemporalPropertyCollection` resource supports retrieving and creating operations via GET and POST HTTP methods respectively.

1. Retrieving operation returns a list of the available abbreviated copy of TemporalProperty object in the specified moving feature.
2. Creating operation post a new TemporalProperty object to the `MovingFeature` that specified by {mFeatureId}.

9.7.1. Operation

Retrieve

1. Issue a `GET` request on the `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties` path

Requirement 33	/req/movingfeatures/tproperties-collection-op/get
A	For every moving feature identified in the MovingFeatures GET response (path `{root}/collections/{collectionId}/items`), the server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties`
B	The path parameter `collectionId` is each `id` property in the `Collection` GET response where the value of the `itemType` property is specified as `movingfeature`. The path parameter `mFeatureId` is each `id` property in the `MovingFeatures` GET response.

Create

This operation is defined in the `CREATE` conformance class of API-Features. This operation targeted TemporalProperty resource.

1. Issue a `POST` request on `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties` path

Support for HTTP POST method is required by API-Features.

Requirement 34	/req/movingfeatures/tproperties-collection-op/post
A	The API implementation SHALL comply with the API-Feature `CREATE` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

Requirement 34	/req/movingfeatures/tproperties-collection-op/post
B	The API implementation SHALL comply with the API-Feature CREATE request body requirements http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete.
C	The content of the request body SHALL be based upon the TemporalProperties schema.

The following example adds a new feature (TemporalProperty resource) to the feature created by the Creation a MovingFeature Example. The feature is represented as JSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Create a New TemporalProperty Object Example

```
Client                                                                  Server
  |                                                                       |
  | POST /collections/mfc-1/items/mf-1/tproperties   HTTP/1.1             |
  | Content-Type: application/json                                        |
  |                                                                       |
  | {                                                                     |
  |    "datetimes": [                                                     |
  |       "2011-07-14T22:01:06.000Z",                                     |
  |       "2011-07-14T22:01:07.000Z",                                     |
  |       "2011-07-14T22:01:08.000Z",                                     |
  |    ],                                                                 |
  |    "speed": {                                                         |
  |       "type": "TFloat",                                               |
  |       "form": "KMH",                                                  |
  |       "values": [65.0, 70.0, 80.0],                                   |
  |       "interpolation": "Linear"                                       |
  |    }                                                                  |
  | }                                                                     |
  |---------------------------------------------------------------------->|
  |                                                                       |
  | HTTP/1.1 201 Created                                                  |
  | Location: /collections/mfc-1/items/mf-1/tproperties/tp-1              |
  |<----------------------------------------------------------------------|
```

9.7.2. Response

Retrieve

A successful response to the TemporalPropertyCollection GET is a document that contains the set of TemporalProperty of the moving feature identified by the {mFeatureId} parameter.

Requirement 35	/req/movingfeatures/tproperties-collection-response/get
A	The API implementation SHALL comply with the API-Features - Part 1:Core Features response requirement http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core/fc-response.
B	Each temporal properties object in the response SHALL include the mandatory properties listed in Table 9.

TemporalPropertyCollection GET Response Schema

```
type: object
required:
  - temporalProperties
properties:
  temporalProperties:
    type: array
    items:
      $ref: temporalProperty.yaml
  links:
    type: array
    items:
      $ref: link.yaml
  timeStamp:
    type: string
    format: date-time
  numberMatched:
    type: integer
    minimum: 0
  numberReturned:
    type: integer
    minimum: 0
```

The following JSON payload is an example of a response to an OGC API-Moving Features TemporalPropertyCollection GET operation.

TemporalPropertyCollection GET Example


```
{
  "temporalProperties": [
    {
      "name": "length",
      "type": "TFloat",
      "form": "http://www.qudt.org/qudt/owl/1.0.0/quantity/Length"
    },
    {
      "name": "speed",
      "type": "TFloat",
      "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
}
```


Create

A successful response to the `TemporalPropertyCollection` POST operation is an HTTP status code.

Requirement 36	/req/movingfeatures/tproperties-collection-response/post
A	The API implementation SHALL comply with the API-Feature `CREATE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

9.7.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

9.8. Resource TemporalProperty

9.8.1. Overview

The `TemporalProperty` resource supports retrieving and creating operations via GET and POST HTTP methods respectively.

1. Retrieving operation returns a `TemporalProperty` resource which is included in the `TemporalPropertyCollection` that specified by {tPropertyName}. The `TemporalProperty` resource returned to the response can be limited using the `limit`, `datetime`, and `leaf` parameters.
2. Creating operation post a new temporal value object to the `TemporalPropertyCollection` that specified by {tPropertyName}.

A temporal property object is a collection of dynamic non-spatial attributes and their temporal values with time. An abbreviated copy of this information is returned for each `TemporalProperty` in the `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties` response.

The schema for the temporal property object presented in this clause is an extension of the `TemporalProperty Object` defined in MF-JSON standard. Table 9 defines the set of property that may be used to describe a temporal property.

Table 9. Table of the properties related to the temporal property

Property	Requirement	Description
name	M	An identifier for the resource assigned by an external entity.
type	M	A temporal property type (i.e., one of 'TBool', 'TText', 'TInt', or 'TFloat').
values	O	A sequence of temporal value
form	O	A unit of measure.
description	O	A short description.

Table 10. Table of the properties related to the temporal value

Property	Requirement	Description
datetimes	M	A sequence of monotonic increasing instants.
values	M	A sequence of dynamic value, having the same number of elements as "datetimes".

Property	Requirement	Description
interpolation	M	A predefined type for a dynamic value (i.e., one of 'Discrete', 'Step', 'Linear', or 'Regression').

NOTE The detailed information and requirements for each property are described in the OGC Moving Feature JSON encoding standard.

Requirement 37	/req/movingfeatures/mandatory-tproperties
A	A parametric value object SHALL contain all the mandatory properties listed in Table 9 and Table 10.

9.8.2. Operation

Retrieve

1. Issue a `GET` request on the `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}` path

The {tPropertyName} parameter is the unique identifier for a single temporal property value offered by the API. The list of valid values for `{tPropertyName}` is provided in the `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties` GET response.

Requirement 38	/req/movingfeatures/tproperties-op/get
A	For every temporal properties in a moving feature (path `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties`), the server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertiesName}`
B	The path parameter `collectionId` is each `id` property in the `Collection` GET response where the value of the `itemType` property is specified as `movingfeature`. The path parameter `mFeatureId` is each `id` property in the `MovingFeatures` GET response. `tPropertiesName` is a local identifier of the temporal properties.

Create

This operation is defined in the `CREATE` conformance class of API-Features. This operation targeted TemporalValue object.

1. Issue a `POST` request on

`{root}/collections/{collectionId}/items/{mFeatureId}/tproperties/{tPropertyName}` path

Support for HTTP POST method is required by API-Features.

Requirement 39	**/req/movingfeatures/tproperties-op/post**
A	The API implementation SHALL comply with the API-Feature `CREATE` operation requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
B	The API implementation SHALL comply with the API-Feature `CREATE` request body requirements `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.
C	The content of the request body SHALL be based upon the TemporalValue schema.
D	The latest date-time instance in the temporal value object in TemporalProperty, determined by `tPropertyName`, SHALL be faster than the beginning date-time instance in the temporal value object in the request body.

The following example adds a new feature (TemporalValue object) to the feature created by the Creation a New TemporalProperty Object Example. The feature is represented as JSON. A pseudo-sequence diagram notation is used to illustrate the details of the HTTP communication between the client and the server.

Create a New TemporalValue Object Example

```
Client                                                                  Server
  |                                                                       |
  | POST /collections/mfc-1/items/mf-1/tproperties/speed   HTTP/1.1       |
  | Content-Type: application/json                                        |
  |                                                                       |
  | {                                                                     |
  |    "datetimes": [                                                     |
  |       "2011-07-14T22:01:09.000Z",                                     |
  |       "2011-07-14T22:01:010.000Z",                                    |
  |    ],                                                                 |
  |    "values": [                                                        |
  |       90.0,                                                           |
  |       95.0,                                                           |
  |    ],                                                                 |
  |    "interpolation": "Linear"                                          |
  | }                                                                     |
  |---------------------------------------------------------------------->|
  |                                                                       |
  | HTTP/1.1 201 Created                                                  |
  | Location: /collections/mfc-1/items/mf-1/tproperties/speed             |
  |<----------------------------------------------------------------------|
```

9.8.3. Response

Retrieve

A successful response to the `TemporalProperty` GET operation is a temporal property identified by the `{tPropertyName}` parameter.

Requirement 40	**/req/movingfeatures/tproperties-response/get.**
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The response SHALL only include temporal properties selected by the request with `limit`, `datetime`, and `leaf` parameters.
C	The content of that response SHALL include the parametric value that defined in the response schema.

TemporalProperty Schema (temporalProperty.yaml)

```yaml
type: object
required:
  - name
  - type
properties:
  name:
    type: string
  type:
    type: string
    enum:
      - TBool
      - TText
      - TInt
      - TFloat
      - TImage
  form:
    oneOf:
      - type: string
        format: uri
      - type: string
        minLength: 3
        maxLength: 3
  description:
    type: string
```

TemporalValue Schema (temporalValue.yaml)

```yaml
type: object
required:
  - datetimes
  - values
  - interpolation
properties:
  datetimes:
    type: array
    uniqueItems: true,
    minItems: 2
    items:
      type: string
      format: date-time
  values:
    oneOf:
      - type: number
      - type: string
      - type: boolean
  interpolation:
    type: string
    enum:
      - Discrete
      - Step
      - Linear
      - Regression
```

The following JSON payload is an example of a response to an OGC API-Moving Features `TemporalProperty` GET operation.

TemporalProperty GET Example


```
{
  "temporalProperties": [
    {
      "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",
        "values": [1.0, 2.4, 1.0],
        "interpolation": "Linear",
      },
      "discharge" : {
        "type" : "Measure",
        "form" : "MQS",
        "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",
        "values": ["car", "human"],
        "interpolation": "Discrete",
      }
    }
  ]
}
```


Create

A successful response to the `TemporalProperty` POST operation is an HTTP status code.

Requirement 41	/req/movingfeatures/tproperties-response/post
A	The API implementation SHALL comply with the API-Feature `CREATE` response requirement `http://www.opengis.net/spec/ogcapi-features-4/1.0/req/create-replace-delete`.

9.8.4. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

Chapter 10. General Requirements

10.1. HTTP Response

Each HTTP request shall result in a response that meets the following requirement.

Requirement 42	/req/general/http-response
A	An HTTP operation SHALL return a response which includes a `status code` and an optional `description` elements.
B	If the `status code` is not equal to 200, then the `description` element SHALL be populated.

The YAML schema for these results is provided in HTTP Response Schema.

HTTP Response Schema

```
title: Exception Schema
description: JSON schema for exceptions based on RFC 7807
type: object
required:
  - type
properties:
  type:
    type: string
  title:
    type: string
  status:
    type: integer
  detail:
    type: string
  instance:
    type: string
```

10.2. HTTP Status Codes

Table 11 lists the main HTTP status codes that clients should be prepared to receive. This includes support for specific security schemes or URI redirection. In addition, other error situations may occur in the transport layer outside of the server.

Table 11. Typical HTTP status codes

Status code	Description
200	A successful request.
201	The server has been fulfilled the operation and a new resource has been created.

Status code	Description
202	A successful request, but the response is still being generated. The response will include a `Retry-After` header field giving a recommendation in seconds for the client to retry.
204	A successful request, but the resource has no data resulting from the request. No additional content or message body is provided.
304	An entity tag was provided in the request and the resource has not been changed since the previous request.
308	The server cannot process the data through a synchronous request. The response includes a `Location` header field which contains the URI of the location the result will be available at once the query is complete Asynchronous queries.
400	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
401	The request requires user authentication. The response includes a `WWW-Authenticate` header field containing a challenge applicable to the requested resource.
403	The server understood the request, but is refusing to fulfill it. While status code `401` indicates missing or bad authentication, status code `403` indicates that authentication is not the issue, but the client is not authorised to perform the requested operation on the resource.
404	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
405	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	Content negotiation failed. For example, the `Accept` header submitted in the request did not support any of the media types supported by the server for the requested resource.
413	Request entity too large. For example the query would involve returning more data than the server is capable of processing, the implementation should return a message explaining the query limits imposed by the server implementation.
500	An internal error occurred in the server.

Annex A: Requirements Detail

NOTE	Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that…
Test method:	Inspect…

A.1.2. Requirement 2

Annex B: Abstract Test Suite (Normative)

Annex C: Examples (Informative)

Annex D: Relationship with other OGC/ISO standards (Informative)

This specification is built upon the following OGC/ISO standards. The geometry concept is presented first, followed by the feature concept. Note that a feature is *not* a geometry, but a feature often contains a geometry as one of its attributes. However it is legal to build features without geometry attribute, or with more than one geometry attributes.

D.1. Static geometries, features and accesses

The following standards define static objects, without time-varying properties.

D.1.1. Geometry (ISO 19107)

The ISO 19107, *Geographic information — Spatial schema* standard defines a `GM_Object` base type which is the root of all geometric objects. Some examples of `GM_Object` subtypes are `GM_Point`, `GM_Curve`, `GM_Surface` and `GM_Solid`. A `GM_Object` instance can be regarded as an infinite set of points in a particular coordinate reference system. The standard provides a `GM_CurveInterpolation` code list to identify how those points are computed from a finite set of points. Some interpolation methods listed by ISO 19107 are (non-exhaustive list):

linear

Positions on a straight line between each consecutive pair of control points.

geodesic

Positions on a geodesic curve between each consecutive pair of control points. A geodesic curve is a curve of shortest length. The geodesic shall be determined in the coordinate reference system of the curve.

circularArc3Points

For each set of three consecutive control points, a circular arc passing from the first point through the middle point to the third point. Note: if the three points are co-linear, the circular arc becomes a straight line.

elliptical

For each set of four consecutive control points, an elliptical arc passing from the first point through the middle points in order to the fourth point. Note: if the four points are co-linear, the arc becomes a straight line. If the four points are on the same circle, the arc becomes a circular one.

cubicSpline

The control points are interpolated using initial tangents and cubic polynomials, a form of degree 3 polynomial spline.

The UML below shows the `GM_Object` base type with its operations (e.g. `distance(⋯)` for computing the distance between two geometries). `GM_Curve` (not shown in this UML) is a subtype of

GM_Primitive. All operations assume static objects, without time-varying coordinates or attributes.

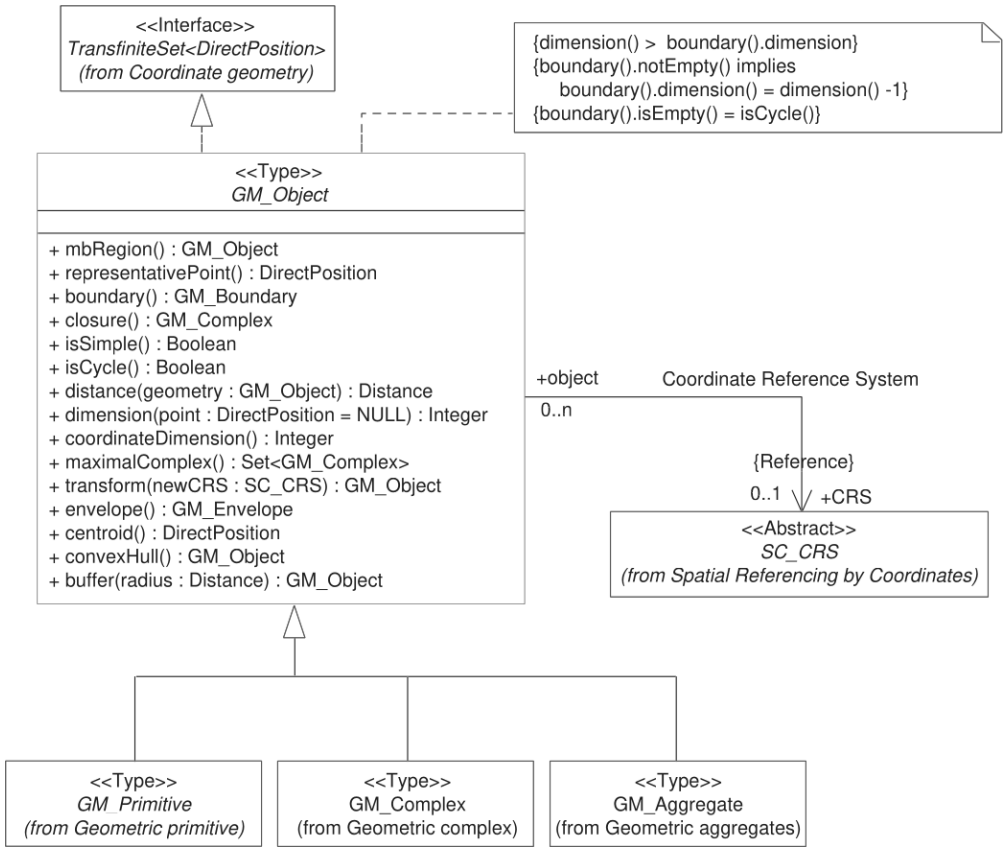

Figure 3. GM_Object from ISO 19107:2003 figure 6

TODO: above discussion is based on ISO 19107:2003. It needs to be updated for latest revisions.

TODO: provide a simplified version of this UML.

Geometry, topology and temporal-objects (GM_Object, TP_Object, TM_Object) are not abstractions of real-world phenomena. These types can provide types for feature properties as described in the next section, but cannot be specialized to features.

D.1.2. Features (ISO 19109)

The ISO 19109, *Geographic information — Rules for application schema* standard defines types for the definition of features. A feature is an abstraction of a real-world phenomena. The terms “feature type” and “feature instance” are used to separate the following concepts of “feature”:

Feature type

The whole collection of real-world phenomena classified in a concept. For example the “bridge” feature type is the abstraction of the collection of all real-world phenomena that is classified into the concept behind the term “bridge”.

Feature instance

A certain occurrence of a feature type. For example “Tower Bridge” feature instance is the abstraction of a certain real-world bridge in London.

In object-oriented modelling, feature types are equivalent to classes and feature instances are equivalent to objects,

The UML below shows the General Feature Model. `FeatureType` is a metaclass that is instantiated as classes that represent individual feature types. A `FeatureType` instance contains the list of properties (attributes, associations and operations) that feature instances of that type can contain. Geometries are properties like any other, without any special treatment. All properties are static, without time-varying values.

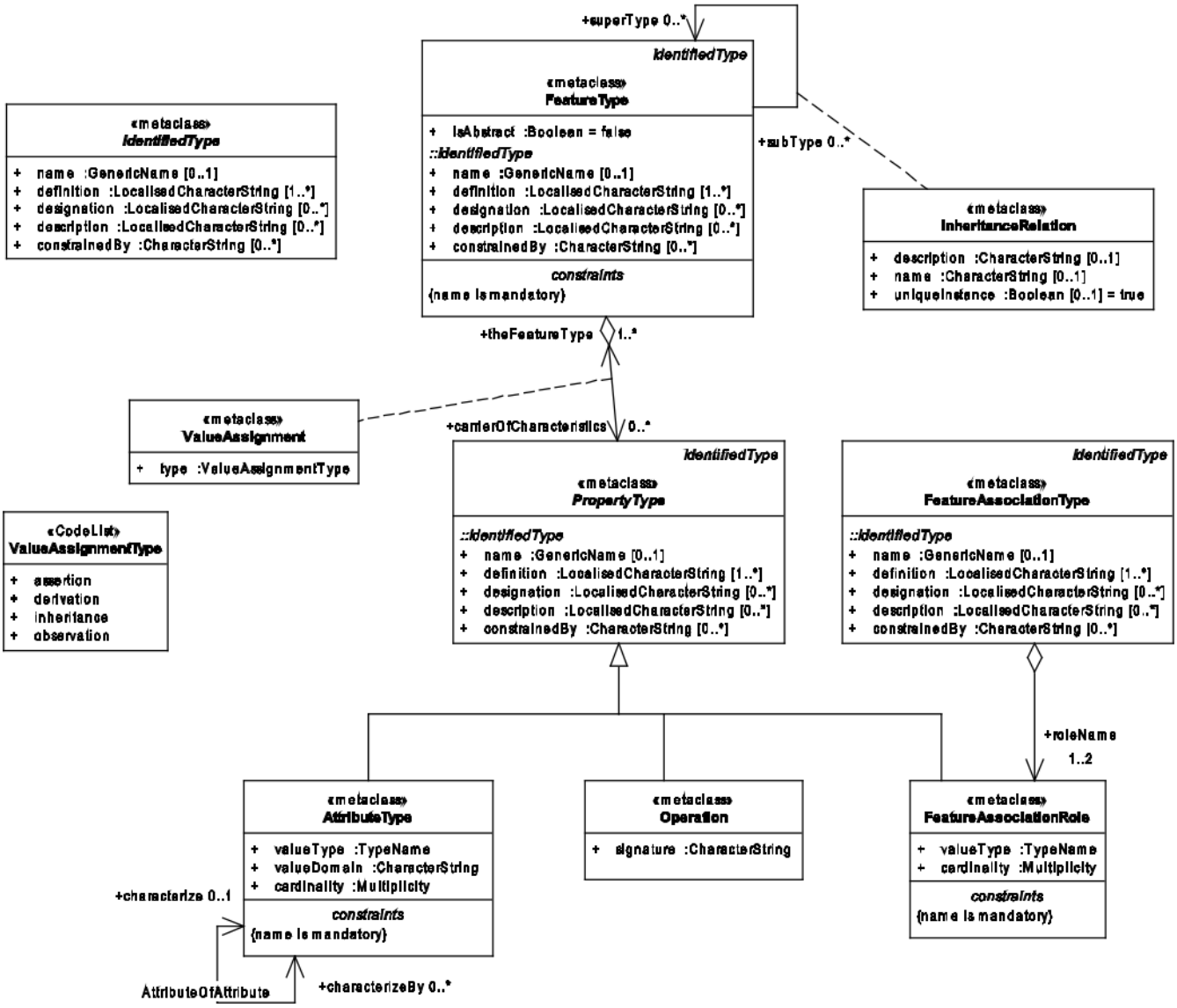

Figure 4. General Feature Model from ISO 19109:2009 figure 5

TODO: provide a simplified version of this UML.

D.1.3. Simple Features SQL

The Simple Feature Access — Part 2: SQL Option standard describes a feature access implementation in SQL based on a profile of ISO 19107. This standard defines *feature table* as a table where the columns represent feature attributes, and the rows represent feature instances. The geometry of a feature is one of its feature attributes.

D.1.4. Filter Encoding (ISO 19143)

The ISO 19143, *Geographic information — Filter encoding* standard (also OGC standard) provides types for constructing queries. These objects can be transformed into a SQL “SELECT … FROM …

WHERE … ORDER BY …” statement to fetch data stored in a SQL-based relational database. Similarly, the same objects can be transformed into an XQuery expression in order to retrieve data from XML document. The UML below shows the objects used for querying a subset based on spatial operations such as “contains” or “intersects”.

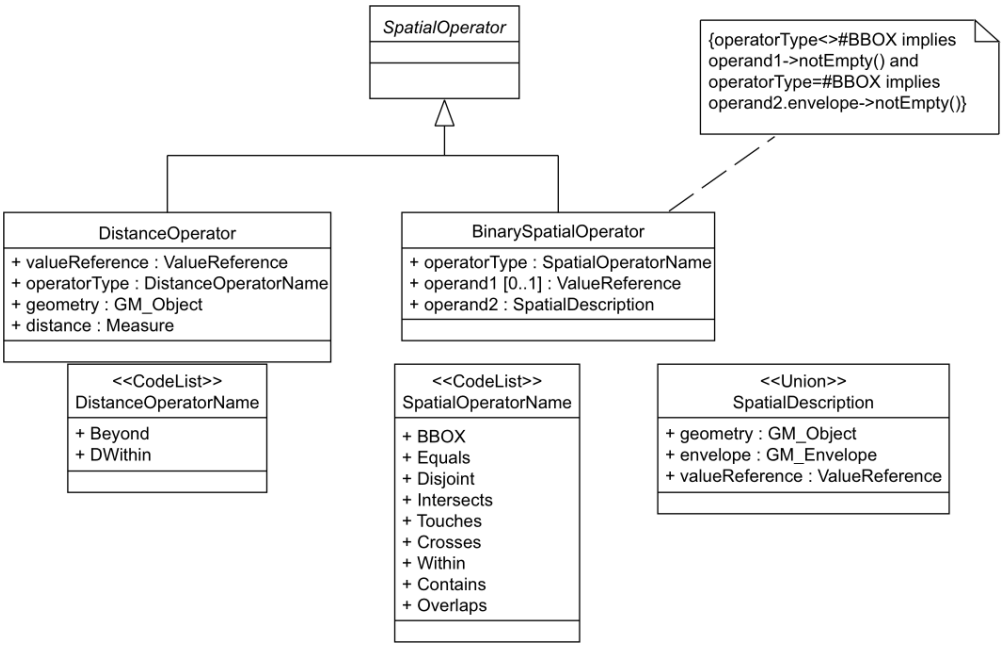

Figure 5. Spatial operators from ISO 19143 figure 6

D.1.5. Features web API

The OGC 17-069, *Features — Part 1: Core* standard specifies the fundamental building blocks for interacting with features using Web API. This base standards allow to get all features available on a server, or to get feature instances by their identifier.

D.1.6. Features Filtering web API

The OGC TBD, *Features — Part 3: Filtering and the Common Query Language (CQL)* standard extends the Feature web API with capabilities to encode more sophisticated queries. The conceptual model is close to ISO 19143.

D.2. Temporal geometries and moving Features

D.2.1. Moving Features (ISO 19141)

The ISO 19141, *Geographic information — Schema for moving features* standard extends the ISO 19107 spatial schema for addressing features whose locations change over time. Despite the “Moving Features” name, that standard is more about “Moving geometries”. The UML below shows how the `MF_Trajectory` type extends the “static” types from ISO 19107.

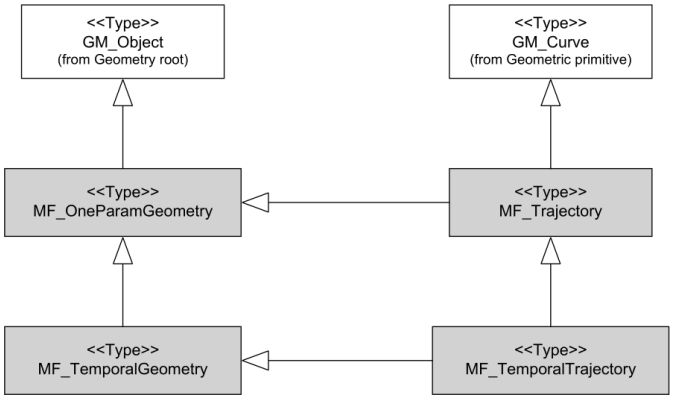

Figure 6. Trajectory type from ISO 19141 figure 3

Trajectory inherits some operations shown below. Those operations are in addition to the operations inherited from `GM_Object`. For example the `distance(⋯)` operation from ISO 19107 is now completed by a `nearestApproach(⋯)` operation.

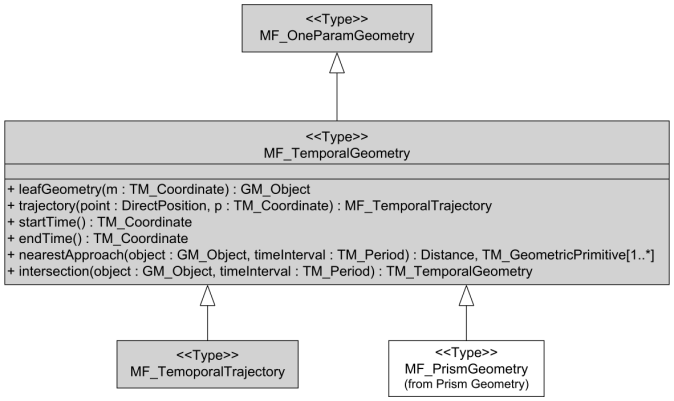

Figure 7. Temporal geometry from ISO 19141 figure 6

D.2.2. Moving Features XML encoding (OGC 18-075)

The OGC 18-075 *Moving Features Encoding Part I: XML Core* standard takes a subset of ISO 19141 specification and encodes it in XML format. But that standard also completes ISO 19141 by allowing to specify attributes whose value change over time. This extension to above *General Feature Model* is shown below:

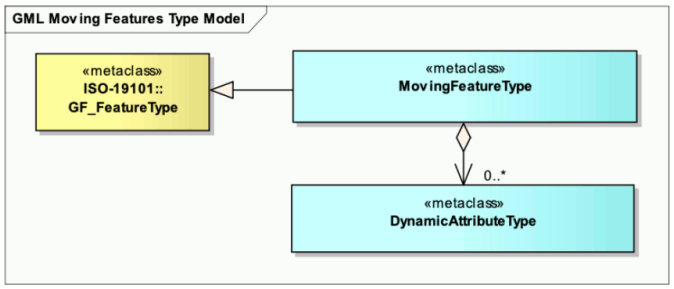

Figure 8. Dynamic attribute from OGC 18-075 figure 3

D.2.3. Moving Features JSON encoding (OGC 19-045)

The OGC 19-045 *Moving Features Encoding Extension — JSON* standard takes a subset of ISO 19141 specification and encodes it in JSON format. The specification provides various UML diagrams summarizing ISO 19141.

D.2.4. Moving Feature Access

The OGC 16-120, *Moving Features Access* standard (TODO)

Annex E: Revision History

Date	Release	Editor	Primary clauses modified	Description
2021-09-14	0.1	Taehoon Kim, Kyoung-Sook Kim, and Martin Desruisseaux	all	first draft version
2022-03-01	0.2	Taehoon Kim, Kyoung-Sook Kim	all	revised sections related to resources to add CRUD operations

Annex F: Bibliography

[1] OGC: OGC Moving Features Encoding Extension - JSON. (2020).

[2] OGC: OGC Moving Features Access. (2017).

[3] OGC: OGC API - Features - Part 1: Core. (2019).

[4] OGC: OGC API - Features - Part 2: Coordinate Reference Systems by Reference. (2020).

[5] OGC: OGC API - Features, https://ogcapi.ogc.org/features/.

[6] OGC: OGC API - Common, https://ogcapi.ogc.org/common/