Open Geospatial Consortium

Submission Date: 2016-10-30

Approval Date: 2017-01-10

Publication Date: 2017-03-12

External identifier of this OGC® document: http://www.opengis.net/doc/is/movingfeatures-access/1.0

Internal reference number of this OGC® document: 16-120r3

Version: 1.0

Category: OGC® Implementation Standard

Editors: Hideki Hayashi, Akinori Asahara, Kyoung-Sook Kim, Ryosuke Shibasaki, Nobuhiro Ishimaru

OGC Moving Features Access

Warning

This document is an OGC Member approved international standard, however, this version is informative. The normative version can be found at: http://docs.opengeospatial.org/is/16-120r3/16-120r3.html This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OGC® Standard
Document subtype: Encoding
Document stage: Approved
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.

Contents

i. Abstract

This document defines Moving Features Access, i.e., access methods to moving feature data for retrieving feature attributes, information on a relation between a trajectory object and one or more geometry objects, and information on a relation between two trajectory objects from a database storing trajectory data of moving features.

Abstract methods of accessing moving features data are defined in ISO 19141:2008 (Geographic information - Schema for moving features) [ISO 19141:2008]. However, the methods are insufficient to access a database storing moving feature data from multiple sources. If implementations for access to moving features data using various programming languages or protocols (e.g., SQL, Java, and HTTP) are developed without any standards, these implementations might be inconsistent with each other, resulting in poor interoperability. Therefore, methods to access a database storing moving feature data are necessary to improve interoperability.

Applications using moving feature data, typically representing vehicles or pedestrians, are rapidly increasing. Innovative applications are expected to require the overlay and integration of moving feature data from different sources to create greater social and business value. Moreover, systems relying on single-source moving feature data are now evolving into more integrated systems. Integration of moving feature data from different sources is a key to developing more innovative and advanced applications.

Moving Features Access ensures better data exchange by handling and integrating moving feature data to broaden the market for geo-spatial information such as Geospatial Big Data Analysis. OGC 14-083r2 (OGC® Moving Features Encoding Part I: XML Core) [OGC 14-083r2] and OGC 14-084r2 (OGC® Moving Features Encoding Extension: Simple Comma Separated Values (CSV)) [OGC 14-084r2] are existing implementation standards. Moving Features Access uses these standards to encode moving features.

ii. Keywords

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

ogcdoc, OGC document, moving features, access method

iii. Preface

This OGC® standard specifies access methods to be implemented for operating against trajectory data of moving features. This specification provides a guideline for implementing interfaces such as SQL functions, Java APIs, and Web APIs.

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):

- Central Research Laboratory, Hitachi, Ltd.
- Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology
- Center for Spatial Information Science, The University of Tokyo
- Defense Systems Company, Hitachi, Ltd.

v. Submitters

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

Name	Affiliation
Hideki Hayashi	Central Research Laboratory, Hitachi, Ltd.
Akinori Asahara	Central Research Laboratory, Hitachi, Ltd.
Kyoung-Sook Kim	Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology
Ryosuke Shibasaki	Center for Spatial Information Science, The University of Tokyo
Nobuhiro Ishimaru	Defense Systems Company, Hitachi, Ltd.

1. Scope

This OGC® Standard specifies abstract methods to access a database storing trajectory data of moving features.

Figure 1 illustrates the concepts of foliation, prism, trajectory, and leaf, which are defined in ISO19141:2008 (Geographic information - Schema for moving features) [ISO 19141:2008]. In this illustration, a 2D rectangle moves and rotates. Each representation of the rectangle at a given time is a leaf. The path traced by each corner point of the rectangle is a trajectory. The set of points contained in all of the leaves, and in all of the trajectories, forms a prism. The set of leaves also forms a foliation. The prism of the moving feature can be viewed as a bundle of trajectories of points on the local representation of the feature's geometry. If viewed in a 4 dimensional spatio-temporal coordinate system, the points on the feature at different times are different points.

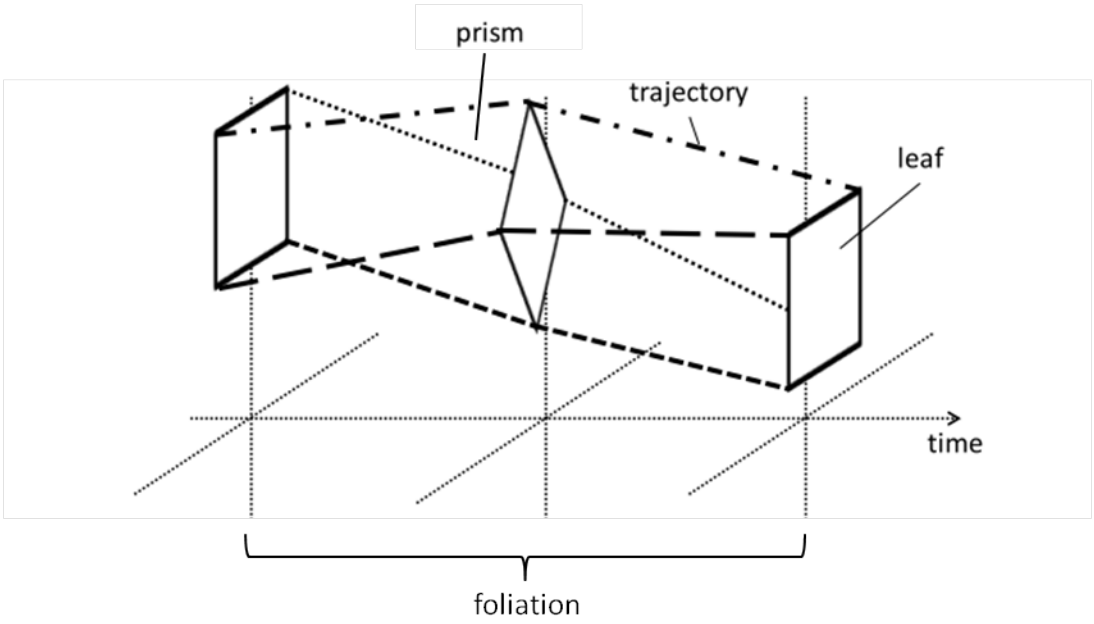

Figure 1 — Foliation, prism and trajectory

Figure 2 shows an example for trajectories of three moving points A, B and C. Each trajectory has a start time and the end time. At t=0 (start of all data), all points start moving. Then, at t =1, the movement of A is changed. In this case, the trajectory of A from t=0 to t=1, the trajectory of A from t=1 to t=2, the trajectory of B from t=0 to t=2, and the trajectory of C from t=0 to t=2 are encoded. This means that only changes of state, including movement speed, direction, existence, and attributes, are encoded in the format. The encoded changes of state are ordered by time in order to determine the states of all features even if only a temporal subset of data is loaded.

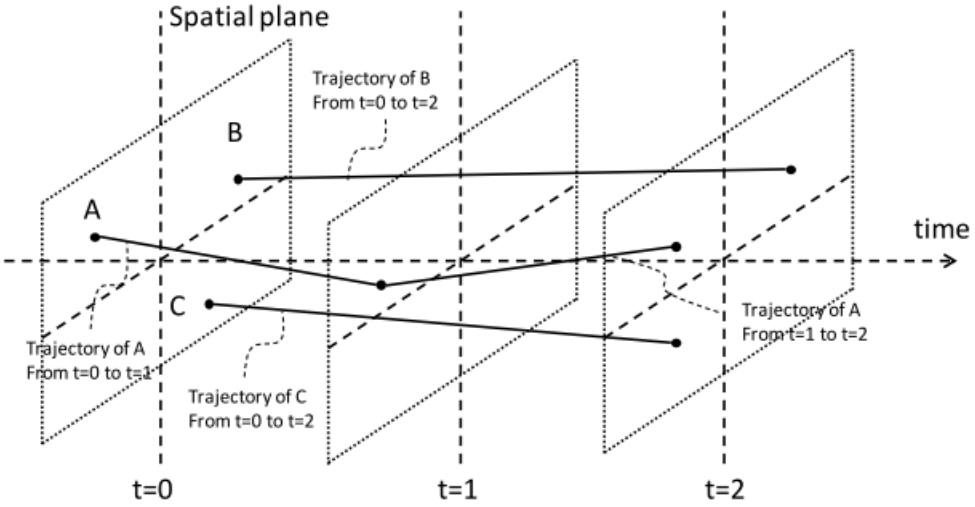

Figure 2 — Example of trajectories

Figure 3 summarizes the operations of existing access standards and Moving Features Access. Operations for geometry objects have been supported by OGC 06-103r4 (OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture) and ISO 13249-3 (Information technology — Database languages — SQL multimedia and application packages Part3: Spatial). Elements of trajectory operations are defined in ISO19141:2008.

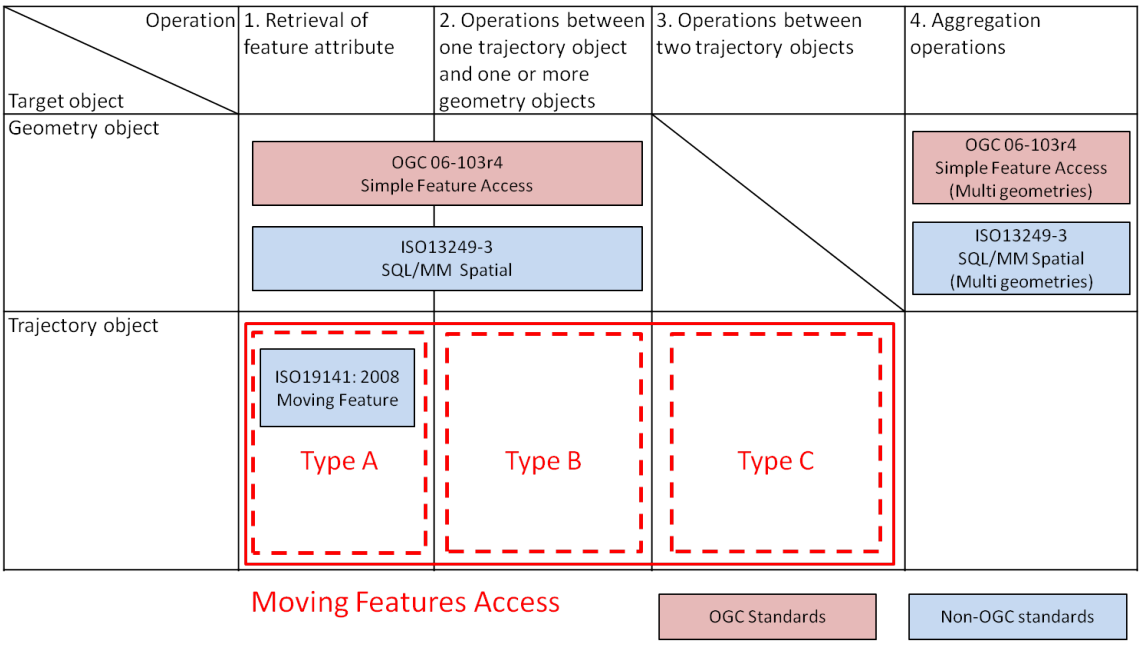

Figure 3 — Existing access standards and Moving Features Access

This standard targets the following three types of operations.

Type A: Retrieval of feature attribute

These operations retrieve positions, trajectories, and velocities of a moving feature such as a car, a person, a vessel, an aircraft, and a hurricane.

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

These operations perform an "intersection" between a geometry object like a administrative boundary and a trajectory of a moving feature like a car, a person, a vessel, an aircraft, and a hurricane.

Type C: Operations between two trajectory objects

An example of these operations is to calculate a distance of the nearest approach of a trajectory to another trajectory. Case studies include calculating the distance between a criminal agent and a police agent for predicting crime patterns or the distance between soccer players for making proper tactics.

This standard does not address all types of operations for trajectory data of moving features. Examples of operations that are out of scope include the following.

Aggregation operations

Examples of these operations are to obtain clusters of multiple trajectories or centroid of multiple trajectories. Such operations will be considered in the future once the demands are clarified.

2. Conformance

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[1].

In order to conform to this OGC™ interface standard, a software implementation shall choose to implement:

a) Any one of the conformance levels specified in Annex A (normative).

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

1 www.opengeospatial.org/cite

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.

OGC: [OGC 14-083r2] OGC® Moving Features Encoding Part I: XML Core, 2015

OGC: [OGC 14-084r2] OGC® Moving Features Encoding Extension: Simple Comma Separated Values (CSV), 2015

ISO / TC 211: [ISO 19141:2008] Geographic information – Schema for moving features, 2008

OGC: [OGC 06-103r4] OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, 2011

ISO: [ISO 13249-3] Information technology — Database languages — SQL multimedia and application packages Part3: Spatial, 2016

OGC: [OGC 06-121r9] OGC Web Services Common Specification, 2010

ISO / TC 211: [ISO 19103:2015] Geographic information – Conceptual schema language, 2015

ISO / TC 211: [ISO 19107:2003] Geographic Information – Spatial schema, 2003

ISO / TC 211: [ISO 19108:2002] Geographic information – Temporal schema, 2002

ISO / TC 211: [ISO 19133:2005] Geographic information – Location-based services – Tracking and navigation, 2005

4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r9], 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.

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

4.1
moving feature
feature whose location changes over time

[ISO 19141:2008]

NOTE: Its base representation uses a local origin and local coordinate vectors, of a geometric object at a given reference time.

4.2
geometric object
spatial object representing a geometric set

[ISO 19107:2003]

4.3
trajectory
path of a moving point described by a one parameter set of points

[ISIO19141:2008]

4.4
one parameter set of geometries
function f from an interval t □ [a, b] such that f(t) is a geometry and for each **point** P □ f(a) there is a one parameter set of points (called the trajectory of P) P(t) : [a, b] →P(t) such that P(t) □ f(t)

[ISO 19141:2008]

4.5
period
one-dimensional **geometric primitive** representing extent in time

[ISO 19141:2008]

4.6
point
0-dimensional **geometric primitive**, representing a position

[ISO 19107:2003]

4.7
geometric primitive
geometric object representing a single, connected, homogeneous element of space

[ISO 19107:2003]

4.8
vector
quantity having direction as well as magnitude

[ISO 19123:2005]

4.9
MF_TemporalTrajectory

An instance of MF_TemporalTrajectory is a **trajectory** whose parameter is time

[ISO 19141:2008]

5. Conventions

5.1 Symbols (and abbreviated terms)

API	Application Program Interface
CSV	Comma Separated Values
CRS	Coordinate Reference System
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
UML	Unified Modeling Language
XML	Extensible Markup Language
2D	Two Dimensional
3D	Three Dimensional

5.2 UML Notation

The diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this standard are described in the diagram below.

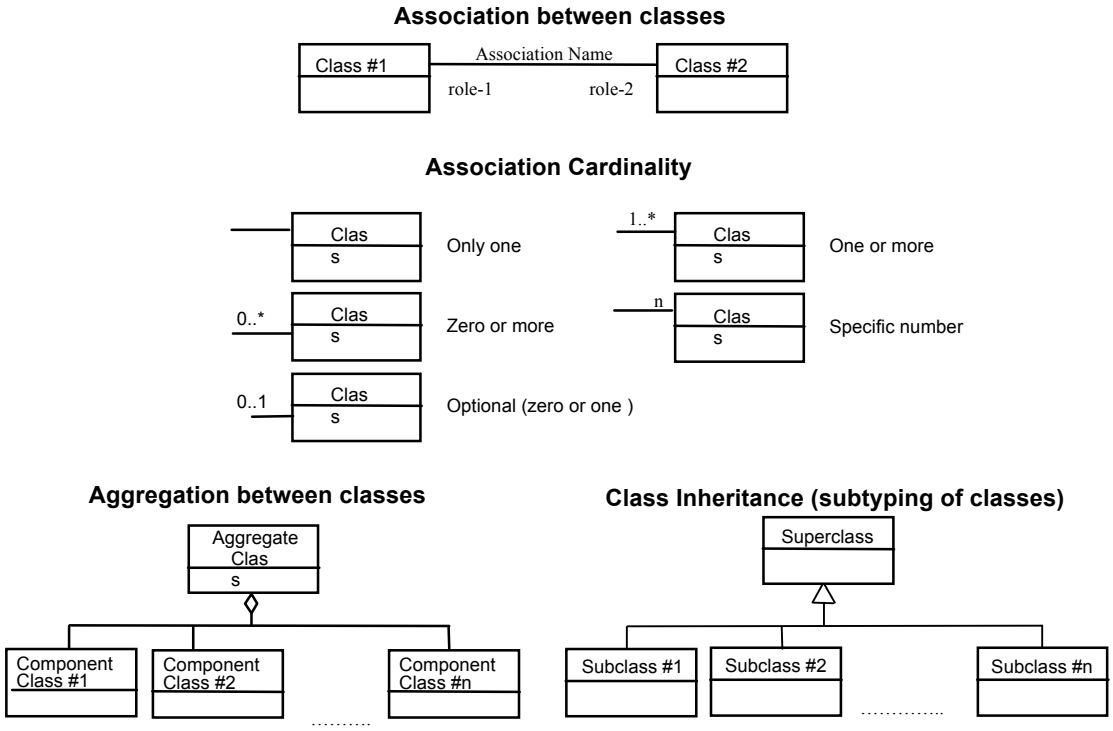

Figure 4 — UML notation

In this standard, the following three stereotypes of UML classes are used:

a) <<Type>> is used to specify a domain of objects together with operations applicable to the objects without defining the physical implementation of those objects. It may also have attributes and associations that are defined solely for the purpose of specifying the behavior of the type's operations and do not represent any actual implementation of state data.

b) <<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

In this standard, the following standard data types are used:

a) Boolean – Only two possible values: TRUE and FALSE

5.3 XML Namespaces

All components of the Moving Feature XML schema are defined in the namespace with the identifier "http://www.opengis.net/movingfeatures/1.0", for which the prefix mf or the default namespace is used within this Standard.

6. Overview

6.1 Operation classes

Moving Features Access is designed to obtain feature attributes, relations between one trajectory object and one or more geometry objects, and relations between two trajectory objects. Hereafter, these are simply called “operations.” This standard specifies the following operations:

Type A: Retrieval of feature attribute;
Type B: Operations between one trajectory object and one or more geometry objects; and
Type C: Operations between two trajectory objects.

6.2 Trajectory data model

The model for trajectory data is defined in the [ISO 19141:2008]. The basic trajectory data model is depicted in Figure 5. Moving Features Access defines methods to operate upon trajectory objects which are instances of MF_TemporalTrajectory.

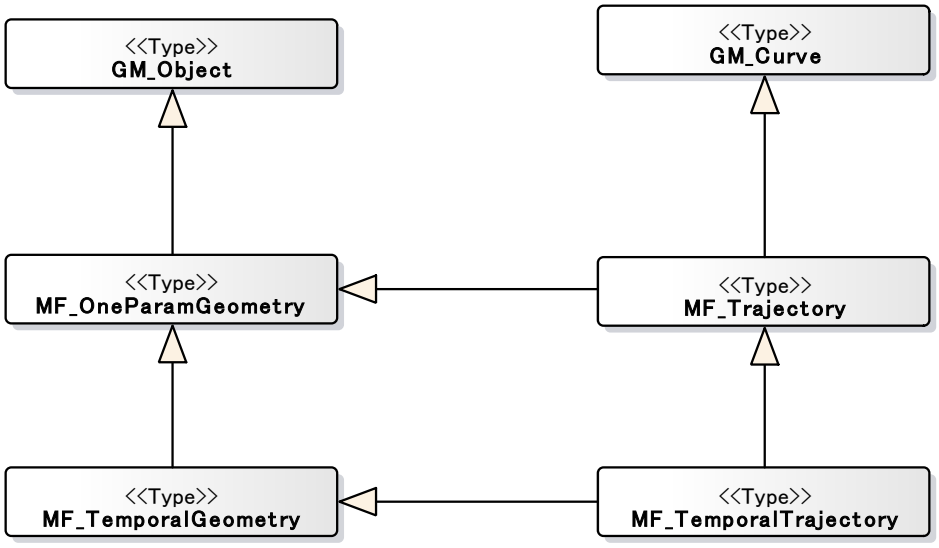

Figure 5 — Trajectory data model in ISO 19141:2008

This standard specifies operations using the trajectory data model in Figure 6. The definitions of Geometry, Curve, and TemporalTrajectory in this figure correspond to those of GM_Object, GM_Curve, and MF_TemporalTrajectory.

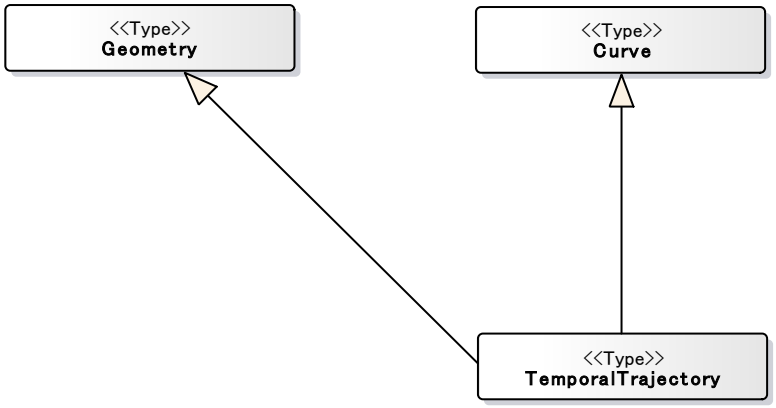

Figure 6 — Trajectory data model in this specification

The operations specified in this standard are defined as methods of TemporalTrajectory as shown in Figure 7. TemporalGeometricPrimitive, TemporalPeriod, TemporalInstant, TemporalCoordinate, and TemporalDuration in this figure correspond to TM_GeometricPrimitive, TM_Period, TM_Instant, TM_Coordinate, and TM_Duration defined in ISO19108:2002 (Geographic information – Temporal schema) [ISO19108:2002], respectively. DirectPosition corresponds to DirectPosition defined in ISO19107:2003 (Geographic Information – Spatial schema) [ISO19107:2003]. LinearReferecePositionExpression corresponds to LR_PositionExpression defined in ISO19133:2005 (Geographic information – Location-based services – Tracking and navigation) [ISO19133:2005]. Distance, Velocity, and Acceleration correspond to Distance, Velocity, and Acceleration defined in ISO19103:2015 (Geographic information - Conceptual schema language) [ISO19103:2015], respectively.

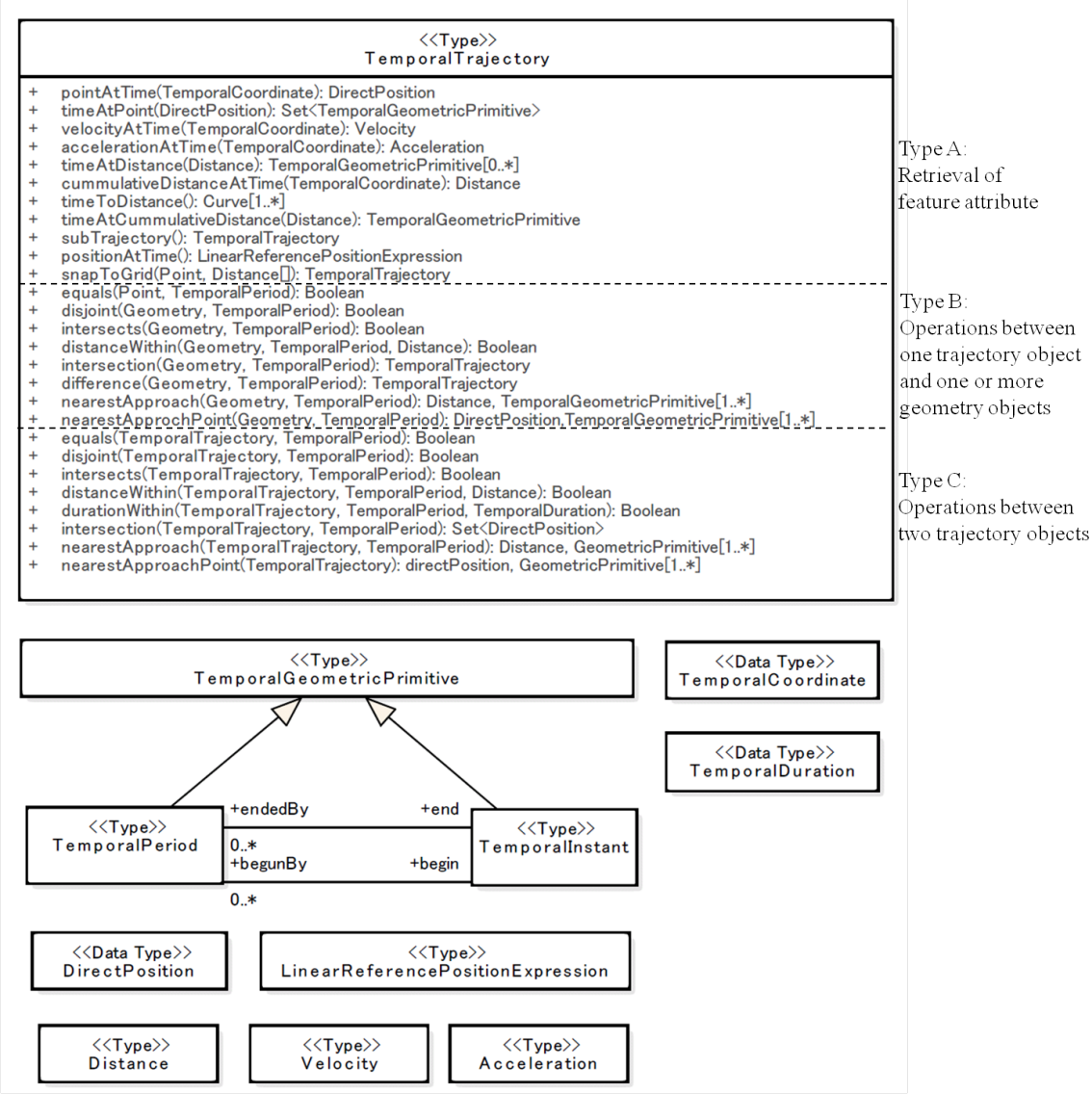

Figure 7 — TemporalTrajectory

7. Type A: Retrieval of feature attribute

The operations on retrieval of a feature attribute are based on operations of TemporalTrajectory.

Req 1 Any implementation of the OGC Moving Features Access SHALL support all requirements listed in Section 7. http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeA

- ***pointAtTime*** **–** shall accept a TemporalCoordinate object in the domain of the TemporalTrajectory object as input, and shall return the DirectPosition object of the TemporalTrajectory object at that time. The DirectPosition object shall hold the coordinates for a position within some CRS.

```
TemporalTrajectory::pointAtTime(t: TemporalCoordinate):
DirectPosition
```

- ***timeAtPoint*** **–** shall accept a DirectPosition object as input and shall return the set of TemporalGeometricPrimitive objects at which the TemporalTrajectory object passes through that DirectPosition object.

```
TemporalTrajectory::timeAtPoint(p: DirectPosition):
Set<TemporalGeometricPrimitive>
```

- ***velocityAtTime*** **–** shall accept a TemporalCoordinate object as input and shall return a Velocity object at that time. The Velocity object shall represent the instantaneous rate of change of position with a time interval and shall hold a unit of measure and a vector.

```
TemporalTrajectory::velocityAtTime(t: TemporalCoordinate): Velocity
```

- ***accelerationAtTime*** **–** shall accept a TemporalCoodinate object as input and shall return an Acceleration object at that time. The Acceleration object shall represent the rate of change of velocity per unit of time and shall hold a unit of measure and a vector.

```
TemporalTrajectory::accelerationAtTime(t: TemporalCoordinate):
Acceleration
```

- ***timeToDistance*** **–** shall return a graph of the time to distance function as a set of curves in the Euclidean space consisting of coordinate pairs of time and distance.

```
TemporalTrajectory::timeToDistance( ): Curve[1..*]
```

Example In Figure 8, the curve represents the time to distance function for an object that accelerates from t_0 to t_1, moves at constant velocity from t_1 until t_2, and then decelerates to a stop at t_3.

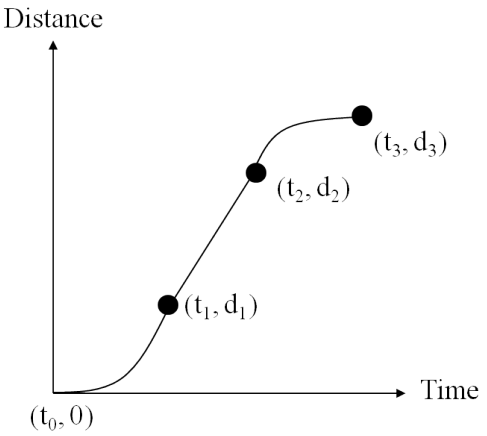

Figure 8 — Example of time to distance curve

- ***timeAtDistance* –** shall return an array of TemporalGeometricPrimitive that lists in ascending order the time or times a particular point (determined by the Set<Distance> in the trajectory's GM_GenericCurve::paramForPoint (p:DirectPosition) : Set<Distance>, DirectPosition) is reached. The Distance object shall hold a unit of measure and a value.

```
TemporalTrajectory::timeAtDistance(d: Distance):
TemporalGeometricPrimitive[0..*]
```

Example Figure 9 shows that timeAtDistance outputs a time instance t when a distance d is input in the case that the time to distance curve is the same with Figure 8.

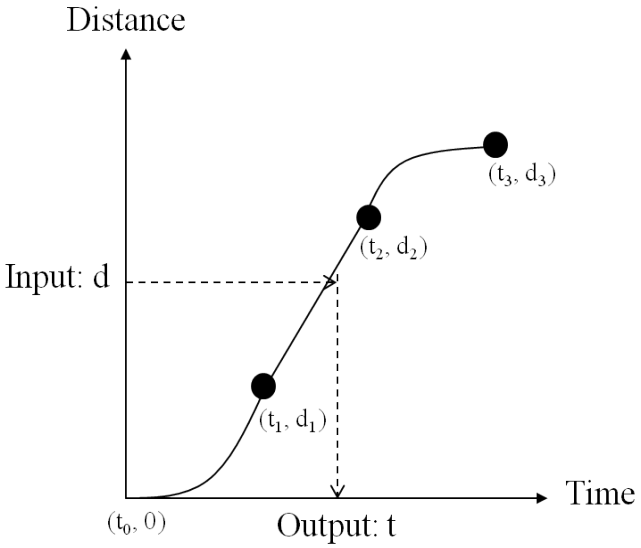

Figure 9 — Example of timeAtDistance

- ***cumulativeDistanceAtTime* –** shall accept a TemporalCoordinate object as input and shall return the cumulative distance traveled (including all movements forward and

retrograde as positive travel distance) from the beginning of the trajectory as a Distance object at that time "t."

```
TemporalTrajectory::cumulativeDistanceAtTime(t: TemporalCoordinate):
Distance
```

- ***timeAtCumulativeDistance*** **–** shall accept a Distance object as input and shall return the time as a TemporalTrajectory object at which the trajectory's total length (including all movements forward and retrograde as positive travel distance) reaches that cumulative travel distance.

```
TemporalTrajectory::timeAtCumulativeDistance(d: Distance):
TemporalGeometricPrimitive
```

- ***subTrajectory*** **–** shall accept two TemporalCoordinate objects in the domain of the trajectory and shall return a TemporalTrajectory object that is a subset of the given trajectory for the specified time interval.

```
TemporalTrajectory::subTrajectory(newStartTime: TemporalCoordinate,
newEndTime: TemporalCoordinate): TemporalTrajectory
```

- ***positionAtTime*** **–** shall accept a TemporalCoordinate object in the domain of the trajectory and shall return the position of the moving feature on the trajectory at that time as a LinearReferencePositionExpression object, expressed as a position in linear reference system. The LinearReferencePositionExpression object describes position given by a measure value, a curvilinear element being measured, and the method of measurement.

```
TemporalTrajectory::positionAtTime(t: TemporalCoordinate):
LinearReferencePositionExpression
```

- ***snapToGrid*** — shall accept a Point object (origin) and an array object including Distance objects whose elements are two cell sizes (x and y cell sizes) or three cell sizes (x, y, and z cell sizes) for a grid as input and shall returns a TemporalTrajectory snapped to the grid (see Figure 10). CRS of the grid is correspondence with CRS of the input Point object. The grid is along axes of the CRS.

```
TemporalTrajectory::snapToGrid(p: Point, cellSize[]: Distance):
TemporalTrajectory
```

Example Figure 10 shows an example of a trajectory object snapped to a grid. In this case, snapToGrid outputs the trajectory shown by the dotted arrow.

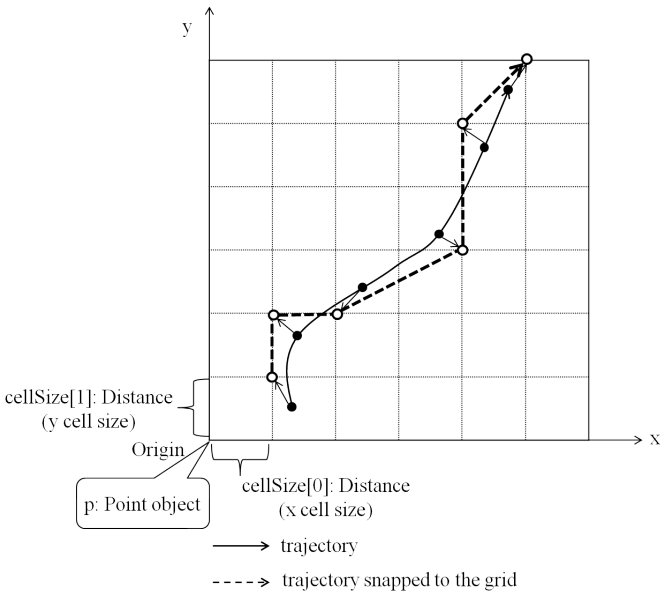

Figure 10 — Example of a trajectory snapped to a grid

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

This section defines operations between one trajectory object and one or more geometry objects. The operations provide spatial relations (e.g., ***intersects***) and spatial statistics (e.g., ***nearestApproach***) between a trajectory object and one or more geometry objects.

Req 2 Any implementation of the OGC Moving Features Access SHALL support all requirements listed in Section 8. http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeB

- ***equals*** **–** shall accept a Point object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object is "spatially equal" to the Point object. The parameter "timeInterval" shall restrict the search to a particular period of time.

  ```
  TemporalTrajectory::equals(p: Point, timeInterval: TemporalPeriod):
  Boolean
  ```

- ***disjoint*** **–** shall accept a Geometry object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object is "spatially disjoint" from the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::disjoint(geometry: Geometry, timeInterval:
TemporalPeriod): Boolean
```

- ***intersects*** **–** shall accept a Geometry object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object "spatially intersects" the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::intersects(geometry: Geometry, timeInterval:
TemporalPeriod): Boolean
```

- ***distanceWithin***— shall accept a Geometry object, a TemporalPeriod object, and a Distance object as input and shall return TRUE if this TemporalTrajectory object and the Geometry object are within the specified distance of one another. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::distanceWithin(geometry: Geometry, timeInterval:
TemporalPeriod, d: Distance): Boolean
```

- ***intersection*** — shall accept a Geometry object and a TemporalPeriod object as input and shall return a TemporalTrajectory object that represents the intersection of this TemporalTrajectory object to the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::intersection(geometry: Geometry, timeInterval:
TemporalPeriod): TemporalTrajectory
```

- ***difference*** — shall accept a Geometry object as input and shall return a TemporalTrajectory object that represents the difference of this TemporalTrajectory object from the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time. The output TemporalTrajectory holds point sets different from the input Geometry object.

```
TemporalTrajectory::difference(geometry: Geometry, timeInterval:
TemporalPeriod): TemporalTrajectory
```

- ***nearestApproach*** — shall accept a Geometry object and a TemporalPeriod object as input and shall return a set of TemporalGeometricPrimitive objects and a Distance object of the nearest approach of this TemporalTrajectory object to the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time. A Distance object output by this operation means a distance between closest pairs of points from the TemporalTrajectory object projected on 2D or 3D space and the Geometry object.

```
TemporalTrajectory::nearestApproach(geometry: Geometry,
timeInterval: TemporalPeriod): Distance,
TemporalGeometricPrimitive[1..*]
```

Example Figure 11 shows that an example of nearestApproach. In this case, a distance output by nearestApproach is the distance between closest pairs of points from the trajectory projected on x-y plane and the geometry object.

- ***nearestApproachPoint*** — shall accept a Geometry object as input and shall return a set of TemporalGeometricPrimitive objects and a DirectPosition object of the nearest approach of this TemporalTrajectory object to the Geometry object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::nearestApproachPoint(geometry: Geometry,
timeInterval: TemporalPeriod): DirectPosition,
TemporalGeometricPrimitive[1..*]
```

Example Figure 11 shows that an example of nearestApproachPoint. In this case, a position output by nearestApproachPoint is position that is projected on x-y plane and is closest to a point of the geometry object.

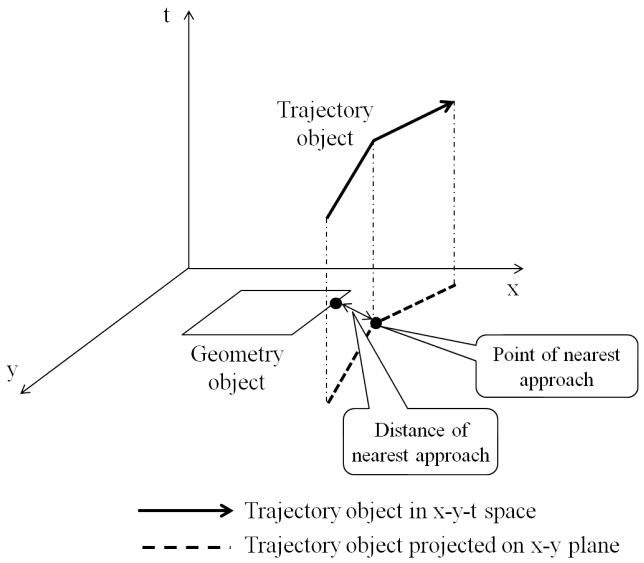

Figure 11 — Example of nearestApproach and nearestApproachPoint between a trajectory and a geometry object

9. Type C: Operations between two trajectory objects

This section defines operations between two trajectory objects. The operations provide spatio-temporal relations (e.g., ***intersects***) and spatio-temporal statistics (e.g., ***nearestApproach***) between two trajectory objects.

Req 3 Any implementation of the OGC Moving Features Access SHALL support all requirements listed in Section 9. http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeC

- ***equals*** **–** shall accept another TemporalTrajectory object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object is "spatially equal" to the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::equals(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod): Boolean
```

- ***disjoint*** **–** shall accept another TemporalTrajectory object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object is "spatially disjoint" from the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::disjoint(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod): Boolean
```

- ***intersects*** **–** shall accept another TemporalTrajectory object and a TemporalPeriod object as input and shall return TRUE if this TemporalTrajectory object "spatially intersects" the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::intersects(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod): Boolean
```

- ***distanceWithin***— shall accept another TemporalTrajectory object, a TemporalPeriod object, and a Distance object as input and shall return TRUE if this TemporalTrajectory object and the other TemporalTrajectory object are within the distance of one another. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::distanceWithin(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod, d: Distance):
Boolean
```

- ***durationWithin***— shall accept another TemporalTrajectory object, a TemporalPeriod object, and a TemporalDuration object as input and shall return TRUE if a difference between a time at which this TemporalTrajectory object passes through a intersection point between this TemporalTrajectory object and the other TemporalTrajectory object and a time at which the other TemporalTrajectory object passes through the intersection point is within the time length.

```
TemporalTrajectory::durationWithin(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod, timeLength:
TemporalDuration): Boolean
```

Example Figure 12 shows that a trajectory object and another trajectory object pass through (x,y) at t_1 and t_2, respectively. If the absolute value of the difference between t_1 and t_2 is below a input time length, durationWithin outputs TRUE.

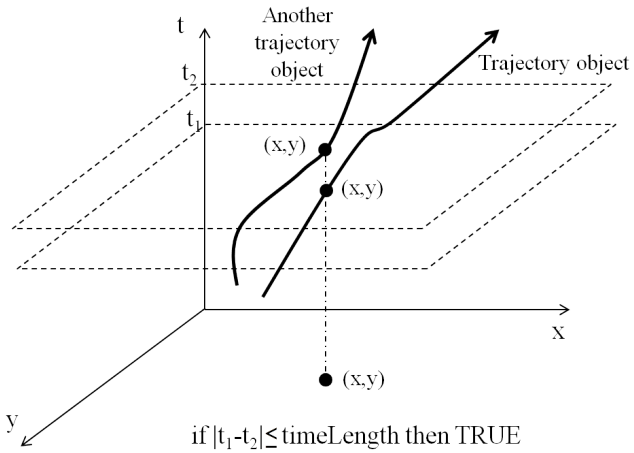

Figure 12 — Example of durationWithin

- ***intersection*** — shall accept another TemporalTrajectory object and a TemporalPeriod object as input and shall return a set of DirectPosition objects that represent the intersection of this TemporalTrajectory object and the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::intersection(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod):
Set<DirectPosition>
```

- ***nearestApproach*** — shall accept another TemporalTrajectory object and a TemporalPeriod object as input and shall return a set of TemporalGeometricPrimitive objects and a Distance object of the nearest approach of this TemporalTrajectory object to the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time. A Distance object output by this operation is a distance between the closest pairs of points from two TemporalTrajectory objects at the same time.

```
TemporalTrajectory::nearestApproach(anotherTemporalTrajectory:
TemporalTrajectory, timeInterval: TemporalPeriod): Distance,
TemporalGeometricPrimitive[1..*]
```

Example Figure 13 shows that a trajectory is nearest approach to another trajectory at t_1. In this case, the distance output by nearestApproach is that between points from these two trajectories at t_1.

- ***nearestApproachPoint*** — shall accept another TemporalTrajectory object and shall return a set of TemporalGeometricPrimitive objects and a directPosition object of the nearest approach of this TemporalTrajectory object to the other TemporalTrajectory object. The parameter "timeInterval" shall restrict the search to a particular period of time.

```
TemporalTrajectory::nearestApproachPoint(anotherTemporalTrajectory:
TemporalTrajectory):directPosition, TemporalGeometricPrimitive[1..*]
```

Example Figure 13 shows a nearest approach from a trajectory object to another trajectory object at t_1. In this case, a position output by nearestApproachPoint is that of the trajectory object at t_1.

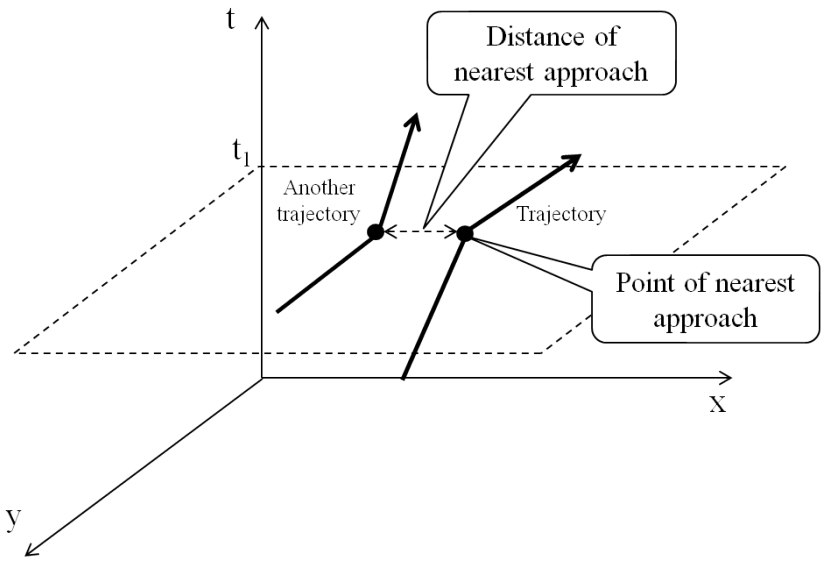

Figure 13 — Example of nearestApproach and nearestApproachPoint between two trajectory objects

10. Exception Guidance

This section describes exception guidance for this standard. Operations of this standard raise the following types of exceptions.

a) Exception caused by invalid operation

This exception type occurs when an operation cannot calculate the output. For example, ***pointAtTime*** cannot return the output if a temporal position of the input TemporalCoordinate object is not included in a time interval of the TemporalTrajectory object. Another example is that ***nearestApproach*** of type C cannot return the output if a time interval of the TemporalTrajectory object is not included in that of another TemporalTrajectory object.

b) Exception caused by a wrong parameter

This exception type occurs when a given parameter has a wrong type or an error in a part of the data format.

c) Exception caused by performance issue

This exception type occurs when a processing time of an operation is very long or when an out-of-memory condition occurs. For example, the processing time of ***nearestApproach*** is very long when the input time interval is very long.

d) Exception caused by an operation not implemented yet

This exception type occurs when an operation is not implemented yet.

Annex A Conformance Class Abstract Test Suite (Normative)

A.1 Introduction

This section describes conformance test for Moving Features Access.

A.2 Test 1: Type A — Retrieval of feature attribute

Test id	http://www.opengis.net/spec/MovingFeatures/Access/1.0/conf/typeA
Requirements	http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeA
Test purpose	Check if any implementation of the OGC Moving Features Access supports all requirements as defined in Section 7.
Test method	Send a query, check if the implementation returns appropriate result as defined in this specification.

A.3 Test 2: Type B — Operations between one trajectory object and one or more geometry objects

Test id	http://www.opengis.net/spec/MovingFeatures/Access/1.0/conf/typeB
Requirements	http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeB
Test purpose	Check if any implementation of the OGC Moving Features Access supports all requirements as defined in Section 8.
Test method	Send a query, check if the implementation returns appropriate result as defined in this specification.

A.4 Test 3: Type C — Operations between two trajectory objects

Test id	http://www.opengis.net/spec/MovingFeatures/Access/1.0/conf/typeC
Requirements	http://www.opengis.net/spec/MovingFeatures/Access/1.0/req/typeC
Test purpose	Check if any implementation of the OGC Moving Features Access supports all requirements as defined in Section 9.
Test method	Send a query, check if the implementation returns appropriate result as defined in this specification.

Annex B The correspondence of concepts of the Moving Features Access with concepts of existing ISO standards (Informative)

B.1 Introduction

This informative annex identifies similarities and differences between Moving Features Access and existing ISO standards.

B.2 Correspondence with ISO 19141:2008

Moving Features Access	ISO 19141:2008	Comment
TemporalTrajectory	MF_TemporalTrajectory	-

B.3 Correspondence with ISO 19103:2015

Moving Features Access	ISO 19103:2015	Comment
Distance	Distance	-
Velocity	Velocity	-
Acceleration	Acceleration	-

B.4 Correspondence with ISO 19107:2003

Moving Features Access	ISO 19107:2003	Comment
Geometry	GM_Object	-
Curve	GM_Curve	-

B.5 Correspondence with ISO 19108:2002

Moving Features Access	ISO 19108:2002	Comment
TemporalGeometricPrimitive	TM_GeometricPrimitive	-
TemporalPeriod	TM_Period	-
TemporalInstant	TM_Instant	-
TemporalCoordinate	TM_Coordinate	-
TemporalDuration	TM_Duration	

B.6 Correspondence with ISO 19133:2005

Moving Features Access	ISO 19133:2005	Comment
LinearReferencePositionExpression	LR_PositionExpression	-

Annex C Implementation Examples (Informative)

C.1 Introduction

This section provides examples of the Moving Features Access operations defined in this standard. RESTful style web services examples are used to describe the operation of Moving Feature Access. Other styles, such as JAVA or C++ could be used to implement the access operations. The examples of access operations refer to OGC 15-078r6 (OGC SensorThings API Part 1: Sensing) [1], which provides an open data model and application programming interface for accessing sensors on the Web by using RESTful approaches and OASIS OData (Open Data Protocol) specification [2].

A client can make an HTTP request to retrieve moving features trajectory data through a simple fixed URL. The operations for trajectory data of moving features are represented by the following URL template format:

GET [service]/MovingFeatures/[version]/[collection]?[operation]{&f=[mime-type]}

The main HTTP methods are POST, GET, PUT, and DELETE corresponding to create, read, update, and delete operations. The Moving Feature Access can be implemented using the HTTP GET method to retrieve feature information and the results of operations between TemporalTrajectory and Geometry/TemporalTrajectory.

- Content surrounded by [] are basic elements and will be replaced with a string literal describing as:
  - service: The URL of service entry point
  - version: The version of OGC Moving Feature Encoding standard
  - collection: A collection of Moving Features; a feature layer
  - operation: Query strings by the use of $select (for Type A) and $filter (for Type B and C) of OData query options with a pre-defined operation name of which is shown in Figure 7
  - mime-type: A media type such as XML(*default*), CSV, JSON, etc.
- Content surrounded by {} is optional.

An Access service returns a response in XML, CSV, or JSON with a status code described by HTTP protocol such as 200 (OK), 400 (Bad Request), or 404 (Not Found). The service returns 200 when the operation was successfully completed. When an exception occurs during the operation, the service returns a status code corresponding to the exception type described in Section 10 as follows:

Table 1 — Exception codes

Exception Type	HTTP Status Code
Exception caused by invalid operation	404
Exception caused by a wrong parameter	406
Exception caused by performance issue	500
Exception caused by an operation not implemented yet	501

C.2 pointAtTime

The following HTTP GET example represents the retrieval of the direct position of the trajectory whose identifier is 1234 at that time in the collection.

```
>> GET
http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles(1234)?$
select=pointAtTime(2013-05-01T10:33:41Z) HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: application/gml+xml
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DirectPosition
xmlns="http://www.opengis.net/gml/3.2">115.86984331920205
40.83836699554286 </DirectPosition>
```

C.3 timeAtPoint

The following HTTP GET example describes a set of times at which each trajectory passed through that direct position in the collection.

```
>> GET http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles?
$select=timeAtPoint(POINT(103%201.0))&f=CSV HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: text/csv
...
id, timeAtPoint
1234, 2013-05-01T10:33:41Z
```

C.4 velocityAtTime

The following HTTP GET example returns an instance of velocity of each trajectory at that time in the collection.

```
>> GET http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles?
$select=velocity(2013-05-01T10:33:50Z)&f=JSON HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: application/json
...
[ {   "id": 1234,
      "velocity": {
            "value": 80,"uom": km/h
       }},
   ...
]
```

C.5 subTrajectory

The following HTTP GET example describes a subset of the trajectory whose identifier is 1234 for the specified time interval.

```
>> GET
http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles(1234)?
$select=subTrajectory(2008-02-05T13:33:00Z, 2008-02-
05T13:34:00Z)&f=GML HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: application/gml+xml
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mf:MovingFeatures xmlns:mf="http://schemas.opengis.net/mf-
core/1.0"
     xmlns:gco="http://www.isotc211.org/2005/gco"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns:gmd="http://www.isotc211.org/2005/gmd"
     xmlns:gts="http://www.isotc211.org/2005/gts"
     xmlns:gml="http://www.opengis.net/gml/3.2"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <mf:stBoundedBy unitOfTime="sec">
    <gml:EnvelopeWithTimePeriod>
      <gml:lowerCorner>116.35971 39.86818</gml:lowerCorner>
       <gml:upperCorner>116.3679 39.86895</gml:upperCorner>
       <gml:beginPosition>2008-02-
05T13:33:00Z</gml:beginPosition>
       <gml:endPosition>2008-02-05T13:34:00Z</gml:endPosition>
     </envelopeWithTimePeriod>
  </stBoundedBy>
  <mf:header></mf:header>
  <mf:foliation>
    <mf:LinearTrajectory gml:id="1234" start="0" end="17">
      <gml:posList>116.38602408293059 39.869248273875 116.3679,
39.86895</gml:posList>
  </mf:LinearTrajectory>
  <mf:LinearTrajectory gml:id="1234" start="17" end="37">
     <gml:posList>116.3679 39.86895 116.36307
39.86857</gml:posList>
  </mf:LinearTrajectory>
  <mf:LinearTrajectory gml:id="1234" start="37" end="47">
     <gml:posList>116.36307 39.86857 116.36195
39.86845</gml:posList>
  </mf:LinearTrajectory>
  ...
 </mf:foliation>
</mf:MovingFeatures>
```

C.6 intersects

The following HTTP GET example filters a set of trajectories that intersect with the parameter geometry object for a particular period of time from the collection.

```
>> GET http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles?
$filter=intersects(POLYGON((30%2010%2C40%2040%2C20%2040%2C10%20
20%2C30%2010)),2013-05-01T10:33:50Z,2013-05-01T10:36:41Z)&f=CSV
HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: text/csv
...
id, intersects
1234, true
2452, true
...
```

C.7 distanceWithin

The following HTTP GET example returns true or false whether trajectory whose identifier is 1234 is located within 100km from the given position during the parameter time period

```
>> GET
http://www.opengis.net/spec/MovingFeatures/1.0/Vehicles(1234)?
$filter=distanceWithin(POINT(103%201.0),2013-05-
01T10:33:50Z,2013-05-01T10:36:41Z,100;km)&f=CSV HTTP/1.1
<<
HTTP/1.1 200 OK
Content-Type: text/csv
...
distanceWithin
true
```

Annex D Revision history

Date	Release	Author	Paragraph modified	Description
2016/9/15	OGC 16-120 r2	Hideki Hayashi	All	Draft for seeking public comment
2016/10/26	OGC 16-120 r3	Hideki Hayashi	Many	Minor edits for TC electronic vote
2017/02/16	OGC 16-120r3	Scott Simmons	Many	Final edits for publishing

Annex E Bibliography

[1] OGC: OGC 15-078r6, OGC SensorThings API Part 1: Sensing, 2016

[2] OASIS: OASIS OData Version 4.0 Part 1: Protocol Plus Errata 02, 2014