Saved 2021-06-16 – 17.30:00

External identifier of this OGC® document: [http: /www.opengis.net/doc/AS/SF/1.0](http://www.opengis.net/doc/AS/SF%20/1.0)

Internal identifier of this OGC® document:
[https:/opengeospatial.org/as/21-011](https://opengeospatial.org/as/21-002.html)r2

Logical identifier of standard
https:/opengeospatial.org/as/SimpleFeatures/2/

Internal reference number of this OGC® document: 21-011r2

Version: 1.0

Category: OGC® Specification

Editor: John R. Herring

21-011r2 Simple Features 2021

Copyright © 2021- Open Geospatial Consortium
to obtain additional rights of use, visit [https:/www.opengeospatial.org/legal/](http://www.opengeospatial.org/legal/).

Warning

This document is a draft 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 nor ISO Standard except as a draft.

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

License Agreement

Per-mission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any Per-son 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 Per-son 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 OPER-ATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR 5655 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 PER-FORMANCE 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

[Simple Features 8](#_Toc71804304)

[1 Scope 8](#_Toc71804305)

[1.1 Normative language 8](#_Toc71804306)

[1.2 Geometry models 9](#_Toc71804307)

[1.3 Feature models 9](#_Toc71804308)

[1.4 General Feature Model (static schema) 9](#_Toc71804309)

[1.5 Dynamic Feature Model 10](#_Toc71804310)

[1.5.1 Dynamic Feature Model (dynamic schema) 10](#_Toc71804311)

[2 Conformance 11](#_Toc71804312)

[2.1 Conformance classes 11](#_Toc71804313)

[2.2 Existing Implementations 14](#_Toc71804314)

[3 Normative references 14](#_Toc71804315)

[4 Definitions 15](#_Toc71804316)

[5 Symbols and abbreviated terms 26](#_Toc71804317)

[6 Feature Taxonomy and Schema 27](#_Toc71804318)

[6.1 Requirements Class: Feature and Property Taxonomies 27](#_Toc71804319)

[6.1.1 Introduction to feature taxonomy 30](#_Toc71804320)

[6.2 Requirements Class: Static Feature Schema 34](#_Toc71804321)

[6.2.1 Feature properties/attributes and associations. 34](#_Toc71804322)

[6.2.2 Feature associations 35](#_Toc71804323)

[6.2.3 Feature Component 35](#_Toc71804324)

[6.2.4 Composite feature and their hierarchies 35](#_Toc71804325)

[6.3 Requirements Class: Dynamic Schema 35](#_Toc71804326)

[6.3.1 Feature types and thematic layering 36](#_Toc71804327)

[6.3.2 Feature associations 38](#_Toc71804328)

[6.3.3 Feature Component 38](#_Toc71804329)

[6.3.4 Object implementation of taxonomy-controlled data 38](#_Toc71804330)

[6.3.5 Relational implementation of taxonomy-controlled data 39](#_Toc71804331)

[6.4 Requirements Class: Ellipsoidal Geometry Model 39](#_Toc71804332)

[6.4.1 Geometry as feature spatial extent 40](#_Toc71804333)

[6.4.2 Coordinate Systems ECEF E3"(X,Y,Z)" and Maps. 40](#_Toc71804334)

[6.4.3 Ellipsoidal Geometry in "(φ,λ)" "(X,Y,Z)" 40](#_Toc71804335)

[6.4.4 Using the radius of curvature as a static or varying radius of curvature 45](#_Toc71804336)

[6.4.5 The Two Options (φ, λ) or (X, Y, Z) Integrals 46](#_Toc71804337)

[6.5 Requirements Class: Topology Query Boolean Relations 49](#_Toc71804338)

[Table 1. Basic Topological Relations 49](#_Toc71804339)

[Annex A Conformance Classes 51](#_Toc71804340)

[A.1 Requirements Class: Feature and Property Taxonomies (Clause 6.1) 51](#_Toc71804341)

[A.2 Conformance Class: 6.2 Requirements Class: Static Feature Schema 53](#_Toc71804342)

[A.3 Conformance Class: Dynamic Feature Schema 54](#_Toc71804343)

[A.4 Ellipsoidal Geometry Model 57](#_Toc71804344)

[Bibliography 58](#_Toc71804345)

Figure

Figures Page

[Figure 1 – Reduced latitude "β", geocentric latitude "ψ" and geodetic latitude "φ" 41](#_Toc71804346)

Table

Table Page

[Table 1 – Example dynamic feature entities in a key-value format similar to Java 30](#_Toc71804347)

1. Abstract

The informative OGC document "Features and geometry – Part 1: Feature models" describes how geographic information in datasets and databases using a "feature model" are structured, created, stored, queried and manipulated.

This standard allows both schemata and ontologies to defined feature models and use the same "linear geometry" that has always.

1. Keywords

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

cartography

controlled vocabulary

geospatial information

geography

taxonomy

ontology

geometry

semantic web

set theory

schemata

dynamic programming languages

geographic

geospatial

database information management

unstructured data

structured data

static programming languages

key-value pairs

1. Preface

This document " Simple Features - 2021" covers:

1. Feature models describe the digital entities in which information is represented and its various logical structures for the representation of real-world phenomena.
2. Schematic, ontological, and taxonomic definitions and representations of features and their properties and relations.
3. Definitions of the structures and operation associated to these digital entities to represent, manipulate and query feature data based on those models. This includes organization for data and database structures such as Relation, Object and NoSQL databases, for object systems either static or dynamic class-based systems.
4. Additional parts will define coordinate reference and geometric systems consistent with geodesy to insure the metric and semantic accuracy of results of the analysis of such data in such models.

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.

1. Submitting members and organizations

The following organizations, members of the OGC Simple Features SWG (standards working group) submitted this Document to the Open Geospatial Consortium (OGC):

Herring, John Oracle

Smyth, Carl Stephen Open Site Plan

Portele, Clemens interactive instruments GmbH

Hayashi, Hideki Hitachi, Ltd., Central Research Lab

[Birkel, Paul](https://portal.ogc.org/?m=users&a=viewuser&user_id=1293) Birkel "Contract Engineer"

and 64 group observers in Simple Features SWG

Simple Features

# Scope

This standard describes how features can be modeled in a 2-dimensional space representing the surface of the geoid (WGS84 for current data) and any derived globe or map derived through a projection, see [38] and [81]. These can display curves and area boundaries modeling geometries which display geographic features of 1-dimension (curves) and 2-dimension (areas) defined by one-dimension boundary curves (which is the essence of ellipsoidal globe, or a map projection from that surface to a planar map). If the application requires elevation, then the uses of latitude (φ), longitude (λ) and elevation (h) can be added.

A map projection cannot suggest common measurable sizes (lengths of curves, areas of polygons in the metric system) unless the projections preserves the latitude and longitude values by a grid system. An ellipsoid are representational shapes but do support metric calculations based on φ "phi" latitude and λ "lambda" longitude coordinates (see clause 6.4, page 39). The map is a projection which is a locally topological equivalent but not metrically equivalent to the ellipsoid, not consistent in geometric measures; See [38] Iliffe & Lott.

Except for the use of the ellipsoid geometric operations defined in clause "6.4. Requirements Class: Ellipsoidal Geometry Model", the topological sections through clause "6. Feature are computational equivalent to earlier "Simple Features" standard with the exception to precise requirements and permissions defined in The Specification Model — A Standard for Modular Specifications (a. k. a. ModSpec).

## Maps, ellipsoids and graticules.

Most of the calculations in this document are on a grid of latitude and longitude on the standard ellipsoid. The calculations for lengths and areas are derived from the graticule and the ellipsoid. On flat maps often have a grid or more precisely a graticule representing the lines of meridians (north-south) and of parallels (east-west). Maps, ellipsoid or data sets with graticules can use all of the equations in this document to calculate linear are areal measures to exact "submeter accuracies" in using the equations below for the WGS84 ellipsoid.

## Normative language

All normative language (statements that directly affect the implementation of items compliant to the standard, called "standardization targets") is restricted to requirements, recommendations, and Permissions. The word "shall" will imply that conformance requires the statement to apply in the conditions cited in that statement; these are referred to as "requirements" and are numbered with "Req #".

The word "should" will imply that the use of best practice recommends the statement to apply in the conditions cited, and these are referred to as "recommendations" and are numbered with "Rec #".

The word "may" will imply that conformance and practice allow the statement to apply in the conditions cited, and these are referred to as "permissions" and are numbered with "Per #". Permissions are not necessarily complete, in the sense that everything not specifically blocked by a requirement is permissible. Permissions are often used to prevent the over-interpretation of requirements which might block a valid implementation of the requirement as stated. This is akin to the open-world assumption, e.g. all approaches are permitted unless explicitly and unambiguously forbidden by a requirement.

Any similar words used the text body ("must", "might", "will", etc.) are not official normative statements and are not marked as such; they usually refer to a logical result of compliance to one or more normative statements, and often appear in the discussion of normative implications. No unmarked statement is normative although it may be truly implied by the marked normative statements. The statements (requirements, recommendations and permissions) marked and numbered are normative even if the "normative language" is inappropriately used.

Requirements ("Req #") are collected in requirements classes. Corresponding conformance classes, in Annex A, collate all requirements associated to each conformance class which outline or suggest tests that may be use in to prove conformance with the associated requirements class. Each requirement class and conformance class have a label in the namespace of this document. Conformance classes often require conformances to a previously defined class.

Recommendations ("Rec #") and Permissions ("Per #") can be anywhere in the document because they are logically not testable and are not addressed in a Conformance Class "testable items" in any sense. Any test of a requirement that blocks the usage of either any recommendation or permission is invalid. All normative statements, requirements, recommendations, permissions, or tests are labelled statements a having a URL in the namespace of the document, which in turn is in the namespace of this document defining the standard. Any test suite that references this document should use the URL’s to validate its procedures. All such names are listed immediately after each named item.

## Geometry models

The extent of a feature may be a point, a curve (expressed as a linestring) or an area (delimited by a polygon, with linestrings as boundaries). A simple curve is a sequence of control points where each arc between two points connects a line between the two consecutive points in the coordinate system in use. Any other curve can be transformed to a line string approximation by inserting new control points between existing control points such as the new points are always on the curve in question.

Because all feature geometry represents a set on the geoid and therefore must be able to convert to ellipsoidal coordinates (φ,λ) i.e. (latitude, longitude). The only maps that are in a Euclidean (X,Y) are engineering datums.

For curves this standard uses line strings for curves and polygonal areas.

## Feature models

This standard enumerates the requirements for defining and representing geographic features in information systems, applications, and transmittable data formats. Each curvilinear feature has a geometry on a two-dimensional surface which is mappable to a standard ellipsoid (usually the GRS84 datum) that represents its footprint as curves represented by a connected sequence of short lines, or an area described by a closed set of non-intersecting closed curves delimitating areas. Any boundary segment cannot have both sides be "in" the polygon .

## General Feature Model (static schema)

The General Feature Model was defined by ISO and OGC in cooperation early in both of their operation. This system followed classical object-oriented programming languages as used at the turn of the century. The origins of this sort of system were generally tied to a single or remarkably similar set of applications. Which meant that each new application might often be to be applied on a dataset unless new properties and object types were added.

In this early approach, a feature is structurally defined by a complied object in a static, compiled class, which includes feature properties, and operations for geographic information applications. This document is derived from and extends *ISO 19109***,** which only addresses schema-based data storage defined in UML, but leaves possibilities open for other non-schematic design systems as extensions, such as dynamic object systems, such as JavaScript and other formats derived from scripting languages usually define in sets of key-value pairs. Dynamic feature systems can readily support both static and dynamic features. In all cases, a dynamic feature model can support both static and dynamic features.

### Dynamic Feature Model (dynamic schema)

The use of dynamic schemata allows flexible data collection and can also support static, compiled feature models where features of the same semantic type have static descriptors which do not vary between instances. For example, the TVA (Tennessee Valley Authority), consist of similar dams that controls water levels for navigation and prevent flooding previous common before the creation of 21 hydroelectric and 17 other dams for complete navigation and flood control in the Tennessee River Valley across 6 states. Although all 38 dams were built by the same organization (TVA), each dam has its own properties. All of them support river navigation, and some of the support roads across the dams and others may be also bridges.

In the current ISO 19109 UML, a static model supports a limited set of feature classes for specific applications. Each feature is in one and only one class which defines a fixed set of attributes and associations. the same set of attributes.

Allowing features to use new and existing attributes can support "surprises" based on unusual examples that would normally require changes in the local schemata that were not expected at the time of the schema creation. In a dynamic system any feature can be associated to properties that are essentially, an Oxford dictionary of words describing properties of features even to the point where each instance can be the only instance of its class.

A dynamic model has a list of feature types, and a list of attribute types. In a running system these schemata lists can continuously be modified, thus supporting existing applications as well as the necessities of both new and old applications.

The strict schematic formulations such as might be used in a complied object language or in a query language requiring a predictable format such as SQL in relational databases or any Object-Oriented Databases which depend on a compiled set of object types, tend towards uniformity in the structure of any feature class. This schema approach matches formal abstractions and is useful in controlled applications that depend on the consistency of a strict relational or object-class structures. In this sense, strict schemata are an interoperability barrier between different applications which may share semantic structure but not implementation structures.

The more flexible feature data structures described in this document, will use semantics, taxonomies and ontologies, to interpret a variable structure that match a common taxonomy and constraints but may vary in their structures because two instances in the real world might not be fully describable by a single "attribute template". The approach follows the ideas of a that strictly defines terms such as, in this document, the names of the feature-types, relations and properties but includes restriction in a class only if demanded by the semantics of reality. Such approaches have proven workable in medical, legal and cultural record keeping, see [39] and [50], and its use is not new to geographic information. Feature and attribute catalogs (which are controlled vocabularies) have been in use in digital mapping since the 1980’s, see [17].

Semantically structured data sets aim to represent reality, not the specific needs of a single application. This leads to application independent data stores that can support all applications because they support a data structure that is both flexible and extendable. This shifts the support of interoperability from creating transfer formats, to creating inclusive, flexible, and thereby interoperable data stores usable by a wide range of application.

# Conformance

## Conformance classes

A "conformance class" evaluates the ability of a system to conform to the mandatory requirements expressed in this document. Recommendations and Permissions are not tested. The Conformance tests are in Annex A where requirements are associated to required tests which for representing feature models types and suppling suggestion to how the tests may be formulated:

1. Feature Schema Conformance (static schema and limited application systems object models)
2. Feature Taxonomy Conformance (taxonomy lists for feature and attribute names that use dynamic languages)
3. Geometry types will be polygonal lines and polygonal areas.
4. Metrics algorithms for length of curves and area of polygons

In a controlled vocabulary, two lists of "well-known" terms are created to act as named keys and searchable indexes for both:

1. Named Feature Types, (feature classes)
2. Named Property Types (attributes
3. Named Feature associations with named roles,

Feature types are a list of feature identities that match one or more feature type names. For example, weir, dam, diversion dam, cofferdam, storage dam, hydroelectric dam, and others "classes". The point is that "dam" is only a root class, with technical differences that suggest subtypes that may change stored attributes and relations, to the extent that each instance of "dam" may be the only member of its subclass.

This can create problems between static and dynamic systems. For example, classical object and relational database systems defined and structured for a single application usage does not normally support such a complex system where features may be close to one object is also one feature class. Database systems have quite flexible methods to deal with this form of real-world complexity. There is a distinction between data storage capabilities, and application requirements.

Property Types are choices of properties (attribute name and value) or associations (feature or features), that can be associated that are linguistically consistent with the feature instance or components (defined in 4.29).

Depending on the flexibility of the system being uses, a feature association can be implemented as a feature where its roles viewed as "properties" which have a type of feature identity. For example, we often give names to these relations, as in "roads cross at an intersection" where intersection becomes a relationship between the roads that meet there. People often do this by naming these place by their properties, such as Five-Points (as in Manhattan and Huntsville, Alabama) or Three-Rivers (as in Pittsburg (Allegheny, Monongahela, and Ohio Rivers) and California, a braided river section on the Kaweah River north of Lake Kaweah, in the Sequoia forests, a Shangri-La for whitewater).

Example: A bridge may carry a roadway, a railway, a canal or a walking path, so that the bridge’s geometry is likely to be part of several feature types even though it is one physical object (real-world phenomena).

In a Feature Ontology as defined in this document, definitions for features, properties and relationships specify the possibilities for data representations, i.e. the form of the digital feature represent one possible view of the feature concept – see [9], [25], [27], [31], [44] and [68]. A feature data set conformant to an ontology is one in which the feature, properties and relationship are semantically consistent with their definitions in the specified ontology and other constraints. Since these constraints should reflect "real-world" truths, in a sense, a valid taxonomy-structured data set which matches the real-world should pass any valid ontology constraints specified by the ontology. In other words, those constraints are tests for the quality of data collection in its representation of reality. If a collected data set is known to be validly consistent with reality, then a "violation" of an ontological constraint may be a flaw in the constraint, not necessarily in the data collection.

In this document, the terms “feature”, “class”, “category” and “taxon” may be used interchangeably. They are essentially different metaphorical views of the same concepts. In a web view, instances of a feature class are feature “resources” and will often be associated with URI identifiers.

"Whereas current Web content has implemented the separation of content from presentation to a large extent, the Semantic Web aims to externalize the inherent semantics from syntax, structure and other considerations. This has led to a layered characterization of metadata that have been used to capture these various aspects of information", separation of concern leads to two meta-requirements, see [44].

* To enable the abstraction of representation details such as the format and organization of data and capture the information content of the underlying data independent of representational details[[1]](#footnote-2).
* To enable the representation of domain knowledge describing the information domain to which the underlying data belongs[[2]](#footnote-3).

The solution for this separation of semantics and structure is found in the use of both schema and ontologies for the definition of data, see [44], Chapters 5, 6 and 7. Classical structured data formats such as relational or XML data require schemata, but the merging of that data requires the understanding of the underlying semantics. Fluid data structures such as JSON (JavaScript Object Notation) or WKT (well-known text) are based on key-value structures which are dependent on data definition and mostly independent of data structure. The merging of these flexible data formats is dependent on a common set of keys, and a common set of definition for those keys (a common set or mergeable sets of taxonomic metadata).

If the data set is controlled by a taxonomy, each feature instance can incorporate any classes, attributes or associations that are consistent with the semantics of the definitions involved. In this case, each feature instance is essentially its own class, an aggregation of the properties defined by its accumulated edits. “Dynamic” programming languages such as JavaScript (objects represented by JSON) and various forms of LISP, see [64], can support this approach. Even a strict object language can be used to build a dynamic class system using associations in lieu of compile-time linkages. Ontology languages begin with a taxonomy but can also add other semantic restrictions like those in an object model. Object models can be mapped to ontologies, but not all ontologies are as restrictive as a non-dynamic UML model. Even in this case, it would be possible to include a feature relation that interlinks object representing the same physical object.

1. In a non-dynamic object model, if two separate class instances represent the same physical object, then there should be an "alternate representation relation" that links those instances based on that equality.

A feature schema implementation contains definitions for each feature type including the properties that might or must be contained and the relationship in which it might or must participate. Since the items in a schema are the same as the items in an ontology, a feature schema should contain a feature taxonomy (or a reference to a default dictionary).

In cases where the "dictionary definitions" for the names of the features, properties or relations is sufficient to meet the ontology’s requirements, then these definitions will be sufficient. In cases where multiple languages are used in the application community for the ontology, a well-structured multi-lingual cross walk will be necessary to alleviate issues of interoperability within and between multi-lingual environments.

This standard defines conformance for combinations of associated instances as follows:

1. A taxonomy and the features and properties it allows consistent with the taxonomy’s definitions,
2. An ontology is a taxonomy where the entities are consistent with the ontology’s restrictions and
3. A schema is an ontology which fully defines the structures of each feature and property type allowed.
4. The ability to calculate in meters the length of any curve representing the extent of a feature.
5. The ability in calculate in square meters the area of any surface representing the extent of a feature.

A taxonomy lists all feature and property types. A single feature instance can have any number of feature type "tags" and any number of property-value pairs if the semantics of the combination are valid, which will be true if the feature instance reflects the existence and state of the real phenomena that the feature represents.

An ontology consistent with this standard will contain a taxonomy, see **Req 48:**, and include constraints, based on the semantics of the interrelationships of the features and properties. In the best circumstances, the features possible in an ontology will be the same as would be collected by the taxonomy alone which, implying that the listed constraints reflect reality.

A schema consistent with this standard will contain a taxonomy, see **Req 50:**, but schemata consistent with previous standards will not. In most of those cases, it defines feature-type and property-type names consistent with geographic tradition and are consistent with common dictionary entries (e.g., Oxford or Webster’s). A schema is more likely to be associated to a specialized application which requires consistent object structures. Schemata tend to segregate feature types, possibly by theme, however the case where a real-world feature/phenomenon is represented by two separate feature types, depending on the system, the digital representation of this feature may be separated into multiple entity/object classes based on applications implying that the real-world feature is instantiated as a digital entity multiple times. If the system recognizes this case, it has mechanisms to represent the reality:

1. A single feature-identity is used for multiple digital-entities which have separate software instantiations, having different schema representations.
2. In a dynamic object model where a single object/entity can instantiate multiple schema classes, the data instances can be structurally isomorphic to the standard taxonomic model.

## Existing Implementations

Some cloud computer implementations use taxonomies, or their near equivalents to structure, index and search for data. Big Data application literature similarly discusses the use of taxonomies to control the semantic of tags and the integration of structured, semi-structured and unstructured data, see [29] and [27].

In the literature, the "taxonomy to ontology to schema" spectrum discussed in this standard is usually referred to as "unstructured to semi-structured to structured" data. The elements of the feature taxonomy are feature class names, the elements of the property taxonomy are property or association role names. Ontologies may add constraints on data element instances and schemata may add constraints which will include a full object model. These constraints are often most useful in testing data for consistency. Each instance of feature shall have a feature identity to facilitate association specification (see Req 41:).

# Normative references

The following publications contain normative information and requirements on the topics in this standard.

ISO 19101-1:2014 Geographic information — Reference model: Part 1:Fundamentals

ISO 19107:2019 Geographic information — Spatial schema

ISO 19109:2015 Geographic information — Rules for application schema

ISO 19111:2019 Geographic information — Referencing by coordinates

SKOS Simple Knowledge Organization System Reference.

OGC 08-131r3 The Specification Model — A Standard for Modular Specifications (a. k. a. ModSpec)

OGC 17-087r13 Features and geometry Part 1 — Feature models

OGC 19-029 Features and Geometry Part 2 — Metrics

Normative statements in this paper are all consistent with 08-131r3 The Specification Model.

# Definitions

In addition to the list below, any definition in any normative reference in Clause 3 will be acceptable. All Standard English words in either in the Oxford, Webster's dictionary or both, with spelling variations apply.

1. abstract root

common root classifier of a category which is a superclass of any classifier in the category

Note: The class (static or dynamic) in any programming languages is the abstract root of any feature. The "feature" (see 4.17) is the root of real-world objects that have both place and descriptions, feature class and attribute.

Note: In this standard, "geometry" is the (named) abstract root for all geometry objects. For simple features, the basic geometry is the line string either for the boundary of area features, or the centerline of a curve feature.

1. application schema

schema that is specific for a specific algorithm or set of algorithms

Note: A ubiquitous database supports multiple applications which support multiple sets of requirements.

1. bearing or compass bearing (between two points)

angle of direction clockwise (usually in degrees) from north of an initial unit tangent of a line-of-sight from the first point to a second point

Note: Direct north is 0˚ increasing to the right (clockwise), east at 90˚, south at 180˚, west at 270˚ and eventually to north again as 360˚.

Clapham, Christopher; Nicholson, James. The Concise Oxford Dictionary of Mathematics (Oxford Quick Reference) (p. 41). OUP Oxford. Kindle Edition.

Note: When a reference curve is used for bearing, the bearing of the tangent along a geodesic will vary. If a projection is used to map to a plane, the line on the plane is not usually a geodesic, if a Mercator projection is used, the line is a "rhumb line", a line of constant bearing. While this is handy for navigation, it is usually not a “shortest line” between the two points on the Earth. For small extends is it sufficiently close not to make significant difference, but the longer the line the bigger the difference.

Note: Usual 2D measure of bearing can be an angle measured from North clockwise, or a unit tangent vector of the geodesic at the first point. If the coordinate system is spatially 3D, the horizontal bearing angle may also need to a vertical altitude angle to be complete. If the reference curve is parameterized by arc length, then the “derivative” is a unit vector. If another parameterization is used, then the derivative may be normalized ($\vec{T}/\left‖\vec{T}\right‖$) e.g. a unit vector.

1. boundary, ∂(geometry)

set that represents the limit of a geometry, the edges of an area, or the ends of a linestring or other one-dimensional geometry

Note: "Boundary" is most used in the context of geometry, where the set is a collection of points or a collection of objects that represent those points. In other arenas, the term is used metaphorically to describe the transition between an entity and the rest of its domain of discourse.

Note: The common "functional" operator for boundary is ∂(geometry).

See 4.35 interior, ι(geometry)

1. <static> class

semantic description of a set of objects that share the same attributes, operations, methods, relationships, and semantics, which require declaration name data attributes and associations variables with fixed attribute types and associations targets defined by static class definition

Note: Static classes are controlled at compile time for the program. All members of a static class are structurally identical with common attributes with common relations as defined by the "static" class definition.

ISO 19103

1. <dynamic> class

semantic description of a set of objects that may share the similar attributes, operations, methods, relationships, and semantics and may change during a running program

Note: Dynamic classes are controlled or modified at runtime of the program. Since the history of each object may be distinct, instances of a dynamic class follow the same semantics, but may differ in attributes and associations based on the history of a different object.

Note: The handling of dynamic class instances allows a wide variants of collection allowing that is more flexible that can collect “truth” in the description of a single object associated to real world objects are not always modified synchronously.

1. coercion,
type coercions,
implicit type conversion

〈programming〉 automatic conversion of a value in one type to another based on the equivalence of values

Note: Coercion may be implemented as a conversion of the value of the expressed type to an expression of a related type (that shares its semantic value but not its structure). It is sufficient that there exists a conversion function taking the original value that would create an equivalent instance of the target class. If a function protocol requires a real number, then the integer (a subclass of real) value of "1" will be coerced into a real number of equivalent value, i.e., the real number format "1.0". Coercion is an automated ability of all popular programming language.

1. cadastre map

large-scale map showing the boundaries of subdivisions of land, usually with the directions and lengths thereof and the areas of individual tracts, compiled for the purpose of describing and recording ownership

Note: Cadastre maps coverage are normally quite small often sufficiently in acres or hectares.

1. controlled vocabulary
controlled taxonomy

orderly and systematic classification of things; producing an established list of terminology (names, words, or phrases) with associated definitions for use to identify, describe, index, or retrieve information

Note: The vocabulary is "controlled" in its use in the sense that only terms from the list may be used for the intended purpose. In this document, the principal vocabulary terms are the "feature", "property", "relation" and "relation roles" in the taxonomy that controls items in a conformant feature data set, and in geographic information by standards.

[Modified from [32], [33], [39] and [50]]

1. default geographic coordinate (φ, λ)

direct position on the ellipsoid in latitude, longitude (φ, λ), from which all projections the calculate position on the mechanism of mapping latitude

Note: Basic coordinates of position on theWGS84 ellipsoid. The coordinates (φ, λ) are not Euclidean, but the ellipsoid is in a 3-dimensional Euclidean space (X, Y, Z).

Note Unlike spherical coordinates, ellipsoidal latitude is measured on the surface of ellipsoid with respect to the polar axis. Ellipsoidal and spherical longitude in both models, the longitude angle is a rotational difference from the prime meridian.

1. dynamic

capable of changing or of being changed.

Note: "Dynamic" with reference to operating systems and programming languages, the implication is that the system can change while it continues to run. As an example, the total amount of memory available may be defined by the contents of a word within the operating system. If this word can be altered without stopping the system and reloading a fresh copy of the operating system, then it is possible to alter dynamically the total amount of memory on the system.

1. dynamic object

〈programming〉 object instance which creates or exposes members (properties, or relations) as necessary at runtime opposed to compile-time e.g., definition time, assumed to support any operations, properties, or relations as necessary; allowing the creation objects to work with structures defined at runtime that do not necessarily match a static or pre-defined types or object format.

Note: Each dynamic object can be modified at run-time by the program adding or modifying an existing object. Once an object is dynamic, it becomes its own class, which can be "cloned" to create identical class instances. Metaphorically, the object can "learn" anything the program or the user can "teach" it.

Note: Dynamic objects are or can be supported in several programming languages such as C# ("C-sharp"), JavaScript, Visual Basic, Python and even C. The programming language cannot constrict the inventiveness of the programmers.

Note: Dynamically typed languages perform type checking at runtime, while statically typed languages perform type checking at compile time.

Note: https://docs.oracle.com/cd/E57471\_01/bigData.100/extensions\_bdd/src/cext\_transform\_typing.html

See and compare: static object.

1. engineering coordinate system

coordinate reference system based on an engineering datum

Note: An engineering coordinate reference system is usually a 2-dimentional Euclidean coordinate system or possibly 3D, e.g. (X, Y) or (X, Y, Z). In a Euclidean coordinate system, the coordinates are based on linear distances.

Note: A cadastral map is a large-scale map showing the boundaries of subdivisions of land, usually with the directions and lengths thereof and the areas of individual tracts, compiled for the purpose of describing and recording ownership.

See: [ISO/TC 211 Multi-Lingual Glossary of Terms](https://github.com/ISO-TC211/TMG), https://github.com/ISO-TC211/TMG

1. engineering datum

datum describing the relationship in a "rectangular" coordinate system to a local reference

1. ellipsoid

<geodesy> geometric reference surface embedded in the GPS 3D Euclidean space represented by an ellipse rotated about the polar axis

Note: For the Earth, the ellipsoid is biaxial with rotation about the polar axis. On an ellipsoid, latitude and longitude are calculated from the direction of the local surface normal or central angle (direction of gravity) for both latitude φ and longitude λ. Because of its symmetry, the use of an ellipsoid the two latitudes (geodetic and central) are identical.

Note: Since the ellipsoid is a reasonable approximation to the geoid, it can be used to create an approximation of the geometry of the earth surface. For example, the metric of the ellipsoid represents a close approximation for geometric for direction or distance-associated calculations, such as geodesics and bearings. Given a full mathematical model of the geoid, the Riemannian geometry of the geoid could be used, but the complexity of such solutions may not be worth the slight increase in precision. In some applications, even simpler spherical approximations are used (e.g., "Spherical Mercator") with sufficient accuracy for the intended purpose.

1. ellipsoidal geometry

variation of spherical geometry using the "radius of curvature functions" for meridians M(φ) and parallels ρ(φ).

Note: See equations in Clause 6.4.3 Ellipsoidal Geometry in "(φ,λ)" "(X,Y,Z)"**; Eq 1 - Eq 21**.

[4], [5], [6], [19], [36], [51], [57], [58], [67], [75], [82].

1. empty set
∅

<mathematics> set without any elements

1. feature

abstraction of a real-world phenomena

Note: A specific feature instance can contain feature attributes and their values and be connected to other features instances by named associations.

[ISO 19109]

1. feature attribute

description of a characteristic of a feature instance

Note: A feature attributes can be associated to a type or to individual instances. A feature class attribute is associated to all members of specified "static" class. A feature class attribute may be associated to any instance of a dynamic class.

1. feature class (static class)

description of a set of feature instances that share the same attributes, operations, relationships, and semantics

1. feature association

relationship that links instances one feature with instances of a different feature instance

1. feature attribute

named property that is associated to a feature instance

Note: A feature class attribute is associated to all members of a static class.

Note: A feature class attribute may be associated to any instance of any class assuming it is semantically valid.

1. feature instance

object representing a feature with attributes, operations and associations that describe named properties containing values for attributes and associations that describe relations that connect this feature to other features

1. point feature (0-dimensional)

point or separate points containing a finite set of separated points, with an empty boundary. ∂P= [∅](#emptySet).

1. curve feature (1-dimensional)

"C(t)" as a curve feature containing separated points along a linestrings of points and line segments between consecutive, with a boundary of the points at the beginning of ends of the linestring segment. The boundary of C is **∂**C=start points and endpoints

1. area feature (2-dimensional)

defined by its boundary, set of geometry objects that are defined by one or more linestrings that are boundary moving around the area on the left of each closed linestring ∂A=the boundary curves

1. feature type

classified group or groups of entities in a set of features that share the similar attributes, operations, methods, relationships, and semantics regardless of the level.

Note: Various feature classes, subtypes or instances may be in the same taxonomic group, e.g. a dirt road, a city street, or a superhighway are all in the same taxon.

1. feature catalog

〈geographic information〉 catalogue containing definitions and descriptions of the feature types, feature attributes and feature associations occurring in one or more sets of geographic data, together with any feature operations that may be applied

Note: This definition of feature catalogue was originally used for the feature models defined by schemata but is general and thus sufficient to cover taxonomic or ontological definitions of feature models.

Note: This is essentially the root rationale for using an "ontology" which merges the concept of "definition" and "class".

[ISO 19126], [ISO 19110]

1. feature component

subset of a feature associated to geometry is a geometric primitive.

Note: In 2 dimensional or 2½ dimensional (associated elevation above the geoid). Primitives would be points, curves or areas (surfaces).

1. feature schema

〈geographic information〉 collection of definitions and declarations for feature classes, properties and associations

Note: In a static schema, properties and associations are associated to classes and are consistent throughout a class.

Note: In a dynamic schema, any property or association can be used for an property or association consistent to the specific object. Such operations to change the structure of an object in a static schema requires require changes to code off-line.

1. feature instance

instance of a feature with associated attributes and relations consistent with its taxonomy or schemata

1. geodetic latitude, φ, φg, and sometimes φ

〈geodesy, astronomy〉 angle that the normal at a point on the reference ellipsoid makes with the plane of the equator, positive north, negative south

Note: The line that determines geodetic latitude is Per-pedicular to the reference ellipsoid and usually does not pass through the center of the ellipsoid, except along the equator or at the poles. The following are valid for the surface of the ellipsoid, where geodetic "φ=φg" and geocentric "ψ=φc" latitudes and "λ" longitude.

Note: If ψ= φ then they are 0°, or ±90°. The tangent at these angles has value of either 0 or ±∞.

Krakiwsky [45]

1. graticule

lattice of parallels and meridians

Note: A graticule on a map makes it possible to associate points on the map to its (φ,λ) coordinates, thus allowing to calculate geometry length and area by using ellipsoidal geometry.

[38]

1. human geography or anthropogeography

branch of geography for the study of the natural and artificial effects derived from interactions between humans, places, and time within the natural and artificial environments

Note: We currently have little experience to support normative input. The Oxford online dictionary and Dictionary.com have variations in the aspects of the issues.

[43],[71].

1. interior, ι(geometry), *i*(geometry)

the geometry not including the boundary.

Note: The interior of a geometry is all of the geometry except it boundary. $A=i\left(A\right)∪∂\left(a\right)$ and $ι(A)∩∂(A)=∅$. The union of the interior and the boundary is the geometry. The intersection of the boundary and interior is empty.

1. latitude

angle between the northward tangent vector to the surface at the point and the direction of polar axis either above (positive) or below (negative) the equatorial plane

1. line, line-segment

curve defined by two coordinate points and the pointes between the two consecutive points $\left(P\_{o},P\_{1},P\_{2},...,P\_{n}\right)$ which $\left(P\_{n-1},P\_{n}\right)$ are two consecutive points on the line string

Note: The segment contains the points between $P\_{o},P\_{1},P\_{2},...,P\_{n} with ΔP\_{i}=\left|P\_{i}-P\_{i-1}\right|$ where 0<t<1.

1. line-string

curve defined by a sequence of connected lines defined by a list of coordinate points$\left(p\_{0},p\_{1},p\_{2},p\_{3},p\_{4,...,}p\_{n}\right)$defined by pairs of consecutive points defining sequential line-segments in the coordinate system in the defining sequence

Note: The coordinates of the list will be in the local coordinate system of the data in use.

Note: If the coordinates are derived from a map projection, accurate ellipsoidal linear and are measures would be more accurate if the measures were converted to latitude and longitude in the ellipsoidal coordinate system.

1. longitude

angle of the normal vector to the surface at the point either east (positive) or west (negative) of the plane of the prime meridian

Note: The coordinate positions on a line of longitude have the same latitude (φ).

Note: A line of longitude is a circle in (X,Y,Z) parallel to the equator.

1. meridian

lines on the surface of the ellipsoid running from pole to pole, a line of equal latitude

Note: The coordinate positions on a meridian have the same longitude (λ).

Note: A meridian is half of an ellipsoid border on a planar surface rotated from the prime meridian by the angle of its longitude. The latitude of a meridian is 90˚ north to -90˚ south.

[38]

1. numeric (or numerical) integration

methods for calculating the numerical value of a definite [integral](https://en.wikipedia.org/wiki/Integral), used to describe the [numerical solution of differential equations](https://en.wikipedia.org/wiki/Numerical_ordinary_differential_equations) method in calculus to use the values of an integral that may not have analytic nor algebraic method of finding numeric solutions such as the Trapezoid or Simpson’s rule.

Clapham, Christopher; Nicholson, James. Concise Dictionary of Mathematics (Oxford Quick Reference) (pp. 335-336). Oxford University Press Oxford. Kindle Edition.

Note: This paper will universally use the Trapezoid Method for numeric integration, because it works well on a smooth surface (the ellipsoid) to find very accurate

1. object

entity with a well-defined boundary and identity that encapsulates state and behavior

[ISO 19107]

1. object model

logical interface, software or system that is modeled using object-oriented techniques that enables the creation of an architectural software or system model prior to development or programming

Note Document object models (DOM) provide a model for a set of objects that provide models for **dynamic** applications.

Note Component object models (COM) provide a model for a software application to support architecture used in compiled applications' software static components.

[62]

1. ontology

〈semantics〉 formal naming and definition of types, properties and relationships of entities that exist in a domain of discourse

Note: The definitions in a feature ontology include features and feature properties (attributes, associations, and roles). An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse and can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models. See [33].

[1], [2], [25], [27], [32], [33] and [39].

1. parallel

lines on the surface of the ellipsoid parallel to the equator, lines of equal latitude

Note: Geometric relations may be defined. Other relations that cannot be calculated from raw data may have to be supported by lists of n-tuples such as exist in a relational database. As discussed in this text, each n-tuple can be considered as "a feature-instance" of the "relation class".

[38] Iliffe & Lott

1. polygon

figure on a surface defined by a set of straight lines segments in the coordinate system

1. schema

〈modeling〉 representation of a plan or theory in the form of an outline or model.

Note: A schema can be expressed as a plan or theory in the form of an outline or model. The realization can be based on definition of entities in the model or as structures representing those entities

[38] Iliffe & Lott

1. polygon

figure on a surface defined by a set of straight lines segments in the coordinate system

Note: A schema is usually consistent list of class-types, each type having a single structure of attributes and association by class, not necessarily by individual object

1. semantic class

set of entities consistent with a taxonomy definition consisting of a necessary and sufficient set of criteria for an entity being an element of the class.

Note: The set of criteria of a semantic class will be necessary but not sufficient for any of its subclasses. A subclass is necessarily more restrictive than any of its super-classes as they inherit their super-class’s criteria but must distinguish themselves as more specific and non-redundant.

1. separation of concerns

design principle for separating a computer program into distinct sections such that each section addresses a separate concern

[79]

1. spatial object

object used to represent a spatial characteristic of a feature

1. spatial operator

operation or procedure that has at least one spatial parameter in its domain or range

1. static object

〈programming〉 object instance, which is structurally defined at compile time, to support a defined list of operations, attribute properties and relationships

Note: Each instant of a static object of a class cannot be modified at run-time to diverge from any other instance of that same object class. Static objects are complied

1. taxon, class, category, type

classifying entity in a taxonomy consisting of a name, a definition and relations to other such entities

Note: The terms are interchangeable, in the sense that a "taxon" in a taxonomy will become a "class" in an object schema, or a type in a "query language", as appropriate. The most common type of relationship is a hierarchical one listing classes as broader (superclass) or narrower (subclass).

1. taxon (plural taxa)

unit of classification, either a collection of similar items or a collection of similar taxa

1. taxonomy

science or technique of classification, or collection of taxa

Note: In this document, the classifications involved are features and their associated properties, relations and attributes.

1. trapezoid method for numeric integration

numeric algorithm to calculate an integral of a function *f(t):[[a, b]*



Note: As 'n' gets larger and the segments gets smaller the approximation gets more accurate. In general, as ∆φ and ∆λ fall below a quarter degree of latitude .25˚ the values approximate to submeter accuracy. The easiest system is to inserts new t’s between pairs doubling the number of trapezoids each time the ∆t average gets smaller.

Note: The examples in this document will use a variation of the trapezoid methods for both area and length calculations. Simple feature uses only piece-wise linear curves, but the examples can be used for any curve types.

1. ubiquitous data set/database

data store not specifically attached to a single or small set of applications

# Symbols and abbreviated terms

The following abbreviations, acronyms, symbols, and names are important key to understanding the normative or informational content to follow. They are in general known publicly, and readers of this document may need to understand their meaning.

|  |  |
| --- | --- |
| API | Application Programming Interface |
| ATS | Abstract Test Suite |
| CRS | Coordinate Reference System, usually a reference to an instance defined in ISO 19111 |
| GIS | Geographic Information System/Science |
| MBR | Minimum Bounding Region (or Rectangle) |
| ∂ | Boundary of a geometry, ∂A is the boundary of A |
| i, ι | Interior of a geometry, iA is the interior of A, e.g. all of A except its boundary. |
| φ, ϕ, phi, lat  | Latitude usually in degrees unless specified as radians for calculations |
| λ, lambda, long | Longitude usually in radians unless specified as degrees or radians for calculations |

The most common geometric form used in numeric integration is a "rectangle" with respect to the latitude and longitude, but with each parallel is slightly shorter as it moves towards the poles. So the "(φ,λ)" coordinate rectangle is a geometric trapezoid in which the longer longitude line is nearer the equator.

# Feature Taxonomy and Schema

Requirements Class: Feature and Property Taxonomies

A feature is a representation of a real-world entity/object. This idea can trace its definition back to the 1960’s in the definition of Simula (a simulation language), the first object-oriented programming language. In Simula, "digital objects" represented "real-world objects"[[3]](#footnote-4). Since the 1980’s, spatial or geographic information systems have paralleled this same idea, at about the same time as Simula and OOPL (object-oriented programming languages) were first being used. Much of the early standardization of GIS followed the same object-oriented paradigm and defined feature classes as object classes (often in UML or its predecessors, such as OMT, object modeling technique).

The "real-world" is not so neat and clean. Any two real-world objects are distinct, often at a fundamental level and, for that reason, each may have a unique logical instance of their own and must be described in different manners with different properties, and not simply different values for the same property. While feature schemata are the core of many systems and their associated standards, there is another approach that supports schema but allows more flexibility as needed. This approach steps back from structural formalism and uses taxonomy and ontology to define features and prescribes descriptive properties of features in a linguistic manner. In contrast with a schema, where a class is defined by its properties, a taxonomy-based system allows any feature to be associated with any property where the feature and property definitions are consistent with one another according to a given vocabulary. In programming this difference of structure is also found, in this case between programming language. The distinction will be "complied vs. interpreted" or "languages or scripts". A geographic feature taxonomy, ontology or schema will define a vocabulary that will identify the semantics of key-value pairs associated to a feature:

* a set of feature types (a type-name and a definition)
	+ including a "is a type of" relation where a definition may reference a more general type,
		- e.g. "a weir is a type of dam such that"…[[4]](#footnote-5)
* a set of feature properties:
	+ a set of feature attributes (an attribute name, a definition and a data structure or object to represent its value), and
	+ a set of feature relations that associate features to one another (a name, a definition, a list of role names and pointer or reference type to identify the feature in each role).

A feature instance contains:

* a unique local identity, (possibly part of a universal identity scheme such as URN)
* one or more feature type names,
* a list of properties with attribute names and associated values,
* a list of feature relationship names each with a list of roles and the target feature identities for those roles.

A geographic feature schema is a restricted ontology that further defines which properties and relations that can be (optional) or that must be (mandatory) associated to instances of which feature types. All properties not specifically mentioned in this list cannot be part of the structure of a feature of that type

A geographic feature ontology is an unrestricted ontology. below shows how a set of features could be represented and gives some potential examples of a taxonomy for feature type definition that are used in the example.

Classical schema formalism arises from application requirements. For example, a radar-based application would concentrate on the attributes of a feature that affect its radar reflectivity, but a visual-based application would concentrate on the visual attributes of a feature such as color. Radar is not good at detecting color (wrong part of the spectrum), but human eyes are. A radar system would not care what color the Golden Gate Bridge is painted, but a system for visual identification would know "international orange" is a good visual recognition clue; in fact, the "color is used in the aerospace industry to set things apart from their surroundings". See [29]. The use of "international orange" is of interest to a "visual-based" application, but not so much to a "radar reflectivity" application. The "interoperability" advantages of a single database for multiple application are obvious, especially when a single operation uses detection approaching the "DC to Daylight"[[5]](#footnote-6) spectrum range.

Once the decision to create this "common database" the use of a "taxonomy merge" is likely unavoidable. Some features will be only one source dataset, but the majority of the "real-world entities" will have multiple instance features to merge. The first step is the merging of the taxonomies of the disparate datasets, then followed by identifying which of those entities are associated to multiple feature instances, which requires the merging of different object instances from different applications. Easy to do in a taxonomy, difficult to do in schema. Over time, the holes may be filled, and the merged dataset will eventually be an application-independent store that supports all participants in the collections of the data, and the synergistic interoperability of the applications.

The driving purpose of OGC is and has always been making applications interoperable. A taxonomy paradigm for datastores seems to be a rational route to creating synergistic applications that have a single view of the world because they can depend on a merged, complete and flexible taxonomic datastore.

The values of attribute properties will have to be represented in a format consistent with the applications supported. In the textual examples in this standard those types may be in text, such as a JSON, YAML or WKT.

We can represent Wheeler Dam as in Table 1; the "URL #" expressions would be system generated entity ids, which can take the real-world name "Wheeler Dam".

In a schema-controlled feature data set, the properties are defined by the controlling schema. For example, the "logical default" for a dam does not have the properties of a bridge, but in the real world several dams have been designed or modified to include a bridge in its superstructure (such as "Hoover Dam" on the Colorado, and "Wheeler Dam" on the Tennessee). In a parallel case, some but not all dams have hydroelectric properties or flood control mechanism, again optional descriptions are needed to create a complete description dependent on the applications using the data. In a schema-controlled data set, the class for "dam" would have to be able to coexist with "bridge" when it is needed. Considering the complexity of reality and difficulty of an *a priori* schema that could adapt to all possibilities, the ontological approach is more likely to be usable without continuous redefinition. A schema would require potentially dynamic multiple inheritance or instantiation in a dynamic object-oriented language.

The "use of taxonomy" versus the "use of schema" is a shift of emphasis from syntax (the form or structure of the information) to semantics (the meaning of the information). This transition should not be a surprise since from the beginning the great barrier to the flow of information has been format i.e. schema, and so the biggest boon to interoperability is to step over that barrier and shift our application in the same direction that the web is moving, towards semantic solutions. This makes the data easier to understand and thus easier to move, and finally easier to interoperate between functionalities.

Table – Example dynamic feature entities in a key-value format similar to Java

|  |
| --- |
| Object-1:  { Feature: "URL 1" ‼ [some URL id] FeatureType: "Dam" FeatureType: "Bridge" Name: "Wheeler Dam" On: "URL 3" ‼ Tennessee River CenterPoint 34°48'14.99" N, -87°22'32.99" W Geometry: Location ‼ {[Line-string expression] }Object-2:  { Feature: "URL 2" ‼ [some URL-id] FeatureType: "Lock" On: "URL 3" ‼ Tennessee River In: "URL 1" ‼ Wheeler Dam }Object-3:  { Feature: "URL 3" ‼ [some URL-id] FeatureType: River  Name: Tennessee River  } |

### Introduction to feature taxonomy

The problem with digital objects representing real world objects is that, in most schematic modeling, all objects of the same class have the same representational structure. The "real-world" is not as neat and clean. Any two real-world objects are distinct, often at a fundamental level and, for that reason, each may be a unique instance of their own "class", e.g. must be described in different manners with different properties, not simply different values for the same properties. While feature schemata are the core of many systems and their associated standards, there is another approach that supports schema but allows more flexibility as needed. This approach steps back from structural formalism and uses ontology, to define feature types and prescribes descriptive properties of features as related parts of a taxonomy. In contrast with a schema, where a class is defined by its properties, a taxonomy-based system allows any feature to be associated with any property where the feature and property definitions are consistent with one another according to a given taxonomy.

A geographic feature taxonomy, ontology or schema will define a namespace that contains definitions of the keys that will be used to identify the semantics of key-value pairs associated to a feature: a set of feature types (a type-name and a definition) including a "is a type of" relation where a definition may reference a more general type, e.g. "weir" is low "dam" built across a river to raise the level of water upstream or regulate its flow or an **e**nclosure of stakes set in a stream as a trap for fish". It is often that common names have several uses, such as in the case of a "weir" it can be a "dam", or it can mean an "enclosure" with various constructions technics, usually based on size or usage. A feature "type" name can have several meanings based on constructions, usage, or used differently based on locality.

A feature type either in a schema or a taxonomy should

* a set of common purposes.
* a set of feature attributes (an attribute name, a definition and a data structure or object to represent its values), and
* a set of feature relations that associate features to one another (a name, a definition, a list of role-names and pointer or reference type to identify the feature in each role).

A feature instance contains:

* a unique local identity, (possibly part of a universal identity schemata)
* one or more feature type names that may vary depending on local usage (e.g. "weir" or dam)
* a list of properties with attribute names and associated values,
* a list of feature relationship names each with a list of roles and the target feature identities for those roles.

A geographic feature schema is an ontology that further defines which properties and relations that can be (optional) or that must be (mandatory) associated to instances of which feature types. All properties not specifically mentioned in this list cannot be part of the structure of a feature of that type. below shows how a set of features could be represented and gives some potential examples of a taxonomy for feature type definition that are used in the example.

Classical schema formalism arises from application requirements. For example, a radar-based application would concentrate on the attributes of a feature that affect its radar reflectivity, but a visual-based application would concentrate on the visual attributes of a feature such as color. Radar is not good at detecting color (wrong part of the spectrum), but human eyes are. A radar system would not care what color the Golden Gate Bridge is painted, but a system for visual identification would know "international orange" is a good visual recognition clue; in fact, the "color is used in the aerospace industry to set things apart from their surroundings". See [23]. The use of "international orange" is of interest to a "visual-based" application, but not so much to a "radar reflectivity" application. The "interoperability" advantages of a single database for multiple application are obvious, especially when a single operation uses detection approaching the "DC to Daylight" spectrum range.

As different applications require integrated formats and information, it is easier to integrate taxonomies than schemata. In fact, integration of schemata often requires that a taxonomy be constructed to map the semantics of attributes and entity names used in the labels of the various schema. These taxonomies can then be compared to determine how the various application schemata can be integrated into a generic "application-independent" format.

Different applications will often have different schemata for the same types of objects, and different data collection rules driven by these differences. If diverse applications are required for a single operation or project, the interoperability advantages inherent in a single datastore, and a single set of "facts" is obvious and usually essential.

The driving purpose of OGC is and has always been making applications interoperable. A taxonomy paradigm for datastores seems to be a rational route to creating synergistic applications that have a single view of the world because they can depend on a merged, complete and flexible taxonomic datastore. The values of attribute properties will have to be represented in a format consistent with the applications supported. In the textual examples in this standard those types may be in text, such as a JSON, YAML or WKT.

In a taxonomy-controlled feature data set, the features and their properties are defined. The association of feature instances to properties only require that the definitions be consistent with data instances. Feature instances would be associated to attributes and association roles defined in the taxonomy as needed. Some programming languages (e.g. JavaScript and LISP) can handle this directly, and non-dynamic object-oriented languages that can be expressed in UML can be programmed to do so also (see Annex 6.8.1). Taxonomy instances for feature types would specify: a feature type name and definition, feature type inheritance, and mandatory properties.

Each of the "dams" serves as part of a water control system (possibly the main purpose of a "dam"), a link in the road system across a waterway (the main purpose of a "bridge") and a source of hydroelectrical power (the main purpose of a "power plant"). This feature-type combination in a single feature is not universal. Some bridges are not associated to neither flood control nor hydroelectric generation. In New York state, the Robert Moses Niagara Hydroelectric Power Station diverts water from above the Niagara Falls and routes it below the falls while generating power. This hydroelectric power plant is not associated to a dam and occupies shoreline only on the New York side of the Niagara River. It also has a Canadian near twin in the Sir Adam Beck Hydroelectric Power Stations in Ontario, Canada. To minimize erosion at the fall’s face, both Niagara facilities are used to affect the amount of water going over the falls at Niagara. The Wheeler Dam on the Tennessee is also used for mosquito control; Wheeler Lake’s water level is periodically adjusted up or down to disrupt the mosquito life cycle either drying out or washing out mosquito larvae. This extra-model information is important to only some applications, but to those applications they are often essential. In the case of the TVA dams mentioned here, mosquito control is as important as the hydroelectric processes. The Tennessee Valley in the last half of the 19th century was affected by yellow fever sometimes yearly, spread by mosquitos.

1. Each feature or attribute type shall be accompanied by a semantically valid definition stating which real-world object might be represented or described by type.
2. Each feature or attribute type shall be defined by a reference to a standard language-specific dictionaries.
3. Each feature instance shall be indexed by any number of semantically consistent feature types and be associated to and indexed by properties or association roles.
4. In a feature ontology, further restrictions may be included to enforce semantic constraints on the data sets. These constraints may include information about valid property and association roles for individual feature types. Any feature instance may be associated to multiple, semantically consistent feature types.
5. A geographic information taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role.
6. The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a multiple inheritance of feature type).
7. The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct super-classes and any required properties.

Note: In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy, i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.
https:/opengeospatial.org/as/SimpleFeatures/3/taxonomy/Req 32:

Note: Any other property type consistent with the feature definition would be optional in this case. In an ontology extension, constraints could be used to forbid specific properties.

1. The taxonomy entry for a feature type may contain common optional properties for entities of the feature type Any feature instance may reference any number of semantically consistent feature types.
2. Any taxonomy item may be associated to searching for instances of that type of item
3. Any property instance may be used to describe any feature instance if the semantic of the feature and property is logically consistent.

Note: Ontology extensions would be able to pre-specify consistent property-feature pairs.

1. The taxonomy entry for a property type shall contain a name, a definition, default data types able to represent the value of the property.
2. In an instance of a property, the value may be any datatype coercible into the default datatype.
3. Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property or association names and roles.
4. A feature taxonomy shall contain a non-empty set of definitions of real-world phenomena "features" that may be represented as features in datasets consistent with this taxonomy.
5. Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.
6. Theme or thematic layer definitions may be root classifiers for feature type definitions. Root classes in a feature taxonomy may be themes or thematic layers. Any feature type definition may contain any applicable subtype of ("is-a") relationship (subclass) to other feature definition
7. Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.
8. Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy.
9. Any feature instances that reference the same real-world phenomena shall have the same feature identifier.
10. A feature access process shall be able to collect all feature instances that refer to the same real-world entity.
11. A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combinations.

Requirements Class: Static Feature Schema

A feature schema will contain a list of static feature types. This is a restriction to modifications of features. If a feature instance does not have an attribute that was not foreseen by the schema, the schema would have to be modified, which may rework the static schema.

### Feature properties/attributes and associations.

A feature model will always contain a non-empty collection of definitions (taxonomy) that either define or describe the potential properties of "real-world" features. The use of any property type to describe any feature type would be the purview of the collection mechanism and what situational peculiarities may require or preclude the use of the properties in the situation.

1. Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities.
2. A feature property definition shall include a unique property name and a datatype.
3. Each instance of feature shall have a unique object identity.
4. Each instance of feature shall have a feature identity.
5. Each real-world phenomenon may be referenced by multiple feature instances.
6. Any two features labeled with the same feature identity shall reference the same real-world phenomenon.
7. Any two features instances that reference the same real-world phenomenon should where feasibly have the same feature identity.

This last recommendation may be difficult, especially across data sets. Object identities are easier to keep unique if a hierarchical identity namespace system such as URN is used. Global grid systems may be useful in defining feature identity system based on location, but dynamic datums, which recognize a dynamic location issue, may be an encumbrance. Unfortunately, the planet is not as stable as we once thought, and we now know things move even without our permission nor intervention.

1. Each feature shall have at least one feature type.

It will of course be possible to have a feature type called "Unknown" but knowing that we do not know is knowledge worth having (akin to Socratic Ignorance, "I do not think I know what I do not know,").

1. Each feature shall be considered a member of each feature type in its type references and any of their super-types and shall have complete ownership of all its property instances values.

Example: Rivers are often broken into reaches (segments of river with no junctions with other waterway excepts at its endpoints). The name of the river, which applies to all its reaches.

### Feature associations

A feature taxonomy or ontology will contain a collection of definitions that either define or describe the potential associations between "real-world" features.

1. A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.
2. Binary association shall be implemented as roles linking each feature in the relation to the other.

### Feature Component

A feature component is a feature whose geometry is a geometric primitive (definition 4.29). In 2D or 2½ data, the primitives would be points, curves and areas (surfaces). In 3D, solids would be included.

A feature containing multiple types of feature primitive should be decomposed into feature components each containing a single geometric primitive. The original feature should then be represented as an aggregation of its components.

Example: Roads are built in segments, usually intersection to intersection, but highways can then be described as a weak aggregation of road segments, some of which carry multiple “route marker” enumerating the shared highway numbers which only change at segment boundaries.

### Composite feature and their hierarchies

A composite feature is an aggregation of smaller, often simpler, features.

1. A feature hierarchy should consist of shared aggregations of composite features and feature components, terminating at the bottom as feature components.

An ontology includes a taxonomy for feature and property. It may add restrictions, for example, limiting which properties are associated to which features. Some describe taxonomy-controlled data as "unstructured" and describe schema-controlled data as "structured". Having labeled the extremes, they declare the choice as binary, missing the broad middle ground, often called "semi-structured".

1. A feature ontology shall be fully compliant as a feature taxonomy.
2. A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.
3. An ontology may add constraints on feature types and properties

## Requirements Class: Dynamic Schema

A feature schema is a feature taxonomy that has a more complete set of constraints consistent with a classical object model.

1. A dynamic feature schema shall be fully compliant with a feature ontology.
2. A dynamic feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology.

### Feature types and thematic layering

A feature taxonomy will always contain at least two collections of definitions (controlled taxonomies, or controlled vocabularies, see [1], [33], and [39]) that separately define and describe "real-world" features and properties. A feature data set consistent with this feature taxonomy.

1. A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.
2. A taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role associated to the taxonomy by its namespace.
3. In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy, i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.

Note: Any other property type consistent with the feature definition would be optional in this case. In an ontology extension, constraints could be used to forbid specific properties.

1. The taxonomy entry for a feature type may contain common optional properties for entities of the feature type.
2. Any feature instance may reference any number of semantically consistent feature types.
3. Any property instance may be used to describe any feature instance if the semantic of the feature and property is logically consistent.

Note: Ontology extensions would be able to pre-specify consistent property-feature pairs.

1. In an instance of a property, the value may be any datatype coercible into the default datatype.

Note: Linkages could be accomplished by URI associations, but other pointer types may be more appropriate for the representation format used for the taxonomy.

1. If an implementation does not support dynamic objects, then any number of static objects may be associated to the same feature.
2. Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.
3. Theme or thematic layer definitions may be root classifiers for feature type definitions.
4. Root classes in a feature taxonomy may be themes or thematic layers.
5. Any feature type definition may contain any applicable subtype of ("is-a") relationship (subclass) to other feature definitions.
6. Any feature instance may contain properties or be involved in associations consistent with its definition.
7. Any feature instances that reference the same real-world phenomena should have the same feature identifier.
8. A feature access process shall be able to collect all feature instances that refer to the same real-world entity.
9. A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.
10. A feature taxonomy shall contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology.
11. Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities
12. A feature property definition shall include a unique property name and datatype.
13. Each instance of feature shall have a unique object identity.
14. Each instance of feature shall have a feature identity.
15. Each real-world phenomenon may be referenced by multiple feature instances.
16. Two features labeled with the same feature identity shall reference the same real-world phenomenon.
17. Any two features instances that reference the same real-world phenomenon should where feasibly have the same feature identity

This last recommendation may be difficult, especially across data sets. Object identities are easier to keep unique if a hierarchical identity namespace system such as URN is used. Global grid systems may be useful in defining feature identity system based on location, but dynamic datums, which recognize a dynamic location issue, may be an encumbrance. Unfortunately, the planet is not as stable as we once thought, and we now know things move even without our Permission nor intervention.

1. Each feature shall have at least one feature type.

It will of course be possible to have a feature type called "Unknown" but knowing that we do not know is knowledge worth having (akin to Socratic Ignorance, "I do not think I know what I do not know,").

1. Each feature instance should have a property "Type" which contains all feature types which this feature instantiates.
2. Each feature shall be considered a member of each feature type in its type references and any of their supertypes.
3. Each feature instance shall have complete ownership of all its property instances values.
4. If a feature is in a containment hierarchy, each property should be present in the largest element consistent with the semantics.

Example: Rivers are often broken into reaches (segments of river with no junctions with other waterway excepts at it endpoints). The name of the river, which applies to all its reaches, should normally be associated with the largest segment who shares that name.

1. A feature property’s definition may contain or be augmented by restrictions on which feature types it describes.

### Feature associations

A feature taxonomy or ontology will contain a collection of definitions that either define or describe the potential associations between "real-world" features.

1. A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.
2. Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.
3. A "n-ary" relation may be implemented as a "relation feature" with n binary relations named for the roles in the original relation.
4. A feature ontology shall be fully compliant as a feature taxonomy.
5. A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.
6. An ontology may add constraints on feature types and properties.

### Feature Component

A feature component is a feature whose geometry is a geometric primitive. In 2D or 2½ data, the primitives would be points, curves and areas (surfaces). In 3D, solids would be included.

1. A feature containing multiple types of feature primitive should be decomposed into feature components each containing a single geometric primitive. The original feature should then be represented as an aggregation of its components.

Example: Roads are built in segments, usually intersection to intersection, but highways can then be described as a weak aggregation of road segments, some of which carry multiple “route marker” enumerating the shared highway numbers which only change at segment boundaries.

1. A feature hierarchy should consist of shared aggregations of composite features and feature components, terminating at the bottom as feature components.

A feature schema is a feature ontology that has a more complete set of constraints consistent with a classical object model.

1. A feature schema shall be fully compliant with a feature ontology.
2. A feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties, and data types required by the definitions in the ontology.

### Object implementation of taxonomy-controlled data

The discussion above implies that the most natural object language to use to implement taxonomic controlled feature data would have to have a dynamic schema/object model. Since most compiled languages (C++, C#) do not work quite that way, this design model assumes a non-dynamic schema by using inheritance and associations.

The classical object implementation of schema-controlled data uses a separate class for each feature type, with embedded attributes for simple properties and associations and roles, "ISO 19109:2015 Rules for application schema". This alternative model breaks the feature classes into separate components allowing a statically structured language to be dynamic.

### Relational implementation of taxonomy-controlled data

In a strong feature schema, where all feature classes with their properties and associations are fully defined, as in the original "Simple Features for SQL" model, can used a single table for each feature class. Extending SQL for geographic information only requires the definition of a SQL-datatype for geometries, see [70].

If we loosen schema constraints, properties will have to be split off into separate tables to be shared by different feature types through foreign key links to these property tables.

To define the idea, the table design below assumes the other extreme where each property and association role can be associated to any feature type, i.e. where only the feature and property taxonomies are used, and no additional constraints are given. Just to be consistent, the taxonomy in the example is presented as a set of relation tables.

## Requirements Class: Ellipsoidal Geometry Model

The extent of a feature may be a point, a curve, or an area. A simple curve is a sequence of control points where each arc between two points are connected by a line between two consecutive points. Any other curve can be transformed to a line sting approximation by inserting new control points between two pre-existing in the existing "control points".

"Part 2: Metrics for geometry" will describe the distinctions between "Euclidean geometry" in a plane such as might be appropriate for a large-scale map and "geodetic, ellipsoidal and non-Euclidean geometry" in general on a curved surface such that of the Earth. The most common implementation is ellipsoidal geometry using the datum surface, usually a reference oblate ellipsoid of the form in equations Eq 1, Eq 2,Eq 3, Eq 4,…, Eq 12.

Spherical geometry is sufficient for a small area using a local best fit spherical approximation consistent with local curvature. Global use of a single sphere is problematic; for example, geodesics on a sphere close in a single arc (great circles) but on more general ellipsoids, only the meridians and the equator do so. The other issue is that the ellipsoidal coordinates used are based not on the central angle of the radial line to the position, but the tangential angles of the surface, which introduces other subtle and non-uniform errors in "spherical approximations".

"Simple geometry" will describe the simplest mathematical geometry definitions, based on linear interpolation. The first 3 parts are a replacement for **ISO 19125-1: 2004, Geographic information – Simple feature access – Part 1: Common architecture**. For completeness, this part will also include definitions for geodesic and rhumb "line" interpolations which complete the concept of "line" from Euclidean geometry. Linear equations in the CRS match the Cartesian coordinate equations for lines. The concept of rhumb lines match the constant bearing concept in navigation, which are lines in maps using a Mercator projection. While linear curves depend on the coordinate applied to the geography, geodesic and rhumb lines are controlled by the geometry of the reference surface (usually an ellipsoid of the CRS). Geodesics match the concept of the shortest distance curve.

Further parts will describe other geometry and topology structures and a set of parallel standards will define WKT (well-known text) and JSON (JavaScript Object Notation) for both features and geometry. It should be noted that WKT and JSON are dependent on key-value encodings, which can use taxonomies to organize the keys, and in some cases, the values as in code lists.

### Geometry as feature spatial extent

A feature is a representation of a real-world object. For spatial applications, the most important property of a feature is location which will be dealt with in later parts of this standard on geometry, which will be derived from and extend and will contain implementation guidance and suggestions. The corresponding SQL syntax will be defined in ISO/IEC 13249-3: Information technology — SQL Multimedia and Application Packages - Part 3: Spatial. Where possible, the SQL/MM approaches to geometry will be paralleled in this series of standards.

In SQL, a purely schema form is used, and dynamic data approaches belong to the looser database models such as NoSQL which depends less on transaction and more of data size and simplicity (big data approaches, see [10]).

### Coordinate Systems ECEF E3"(X,Y,Z)" and Maps.

There are several coordinate systems used in geographic information for calculating accurate measures of distance (curve lengths in meters) and area (polygon areas in square meters). Mathematically, the usual position is an ellipsoidal position in latitude and longitude (φ, λ), and height above the ellipsoid along the local perpendicular (*h*). The ellipsoid is approximation of the geoid, which is the shape that a water surface would approximate under the influence of the gravity and rotation of Earth, if other influences were absent, and differs from the ellipsoid in height between +85m to -107m at most, see [38] and [74]. Maps are two-dimensional projections from the ellipsoid.

As it will be seen, the calculations of distances and areas need to be made on the WGS84 ellipsoid. The coordinates (φ,λ) are non-Euclidean marking the ellipsoid. The ellipsoid is embedded 3-dimensional Euclidean space (X,Y,Z). This Earth-Centered, Earth Fixed Euclidean (ECEF E3) is a standard Cartesian 3 space E3 centered at (0,0.0) with (X,Y,Z) units in meters. The X-axis passes through the prime meridian and the equator at lat-long (φ,λ)=(0,0) and at (X,Y,Z)=(6378137,0,0). The positive Y-axis goes through (X,Y,Z)=(0,6378137,0) at (φ,λ)=(0,+90∘).

Satellite Navigation systems work in this coordinate system by measuring the distance from a set satellites (with known positions) and measure distance by comparing a time signal from the satellite with the local time (e.g. the time delay determines the distance from the satellite transmitter to the receiver). See [30].

### Ellipsoidal Geometry in "(φ,λ)" "(X,Y,Z)"

The coordinates of the ellipsoidal surface in latitude φ and longitude λ is in general considered a best fit "spheroid" surface that is consistent with the concepts of latitude and longitude. Latitude is the angular height of the of the pole (about 3° 8' away from the star Polaris), and longitude is the angular differential from the 0° latitude from the Prime Meridian.

The following equations describe the important information of the ellipsoids including geodetic coordinates (φ, λ), where "a" is the semi-major axis (equatorial radius = 6,378,137.0 m) and "b" is the semi-minor axis (polar radius = 6,356,752.314245). The inverse flattening is 298.257223563. Although all equations are valid for all ellipsoid, the numerical values are WGS84, the common spheroid for GPS.

The length of angles in either latitude (φ) or longitude (λ) used in any calculation shall be expressed in radians.

The length of curves and area in latitude (φ) or longitude (λ) used in any calculation shall be expressed in meters or square meters.

Ellipsoid (surface) and (solid) parameter. The equations are commonly included in Geodesy texts, and GPS texts; see .



Figure 1 – Reduced latitude "β", geocentric latitude "ψ" and geodetic latitude "φ"

(larger eccentricity for emphasis)

Burkholder [6], Bomford [5] Clynch [13], Hotine [36], IOGP [41], Jekeli [42], Krakiwsky and Thomson [45], Ligas [46], Panigrahi [53], Torge [67]

1. Ellipsoid surface 2D: 
2. Ellipsoid solid 3D: 
3. Ellipsoid equatorial radius: 
4. Ellipsoid polar radius: 
5. First eccentricity: 

Length of the normal at latitude φ, in radians, polar axis to surface (see [5], Table 7.2).

1. Length Point to Axis: 
2. Radius of meridians: 

Note: If calculations require elevation (h) above the ellipsoid (φ, λ) then the value of N(φ) can be replaced by (N(φ)+h).

Radius of curvature on a parallel ρ(φ) and meridian M(φ) (see [6], Table 7.2)

1. Radius of parallel 

Note: A parallel is a circle with a radius of curvature ρ(φ). The radius of curvature of a meridian with "a" along the equator

1. Lat-long (φ,λ) to (X,Y,Z) 

The rectangle below has equal meridian length both in north-south angles (of latitude) and meters. Parallel lengths vary by N-S differences (the parallel that is closer to the equator is longer than one that is nearer to the pole) . A longitude length in meters differ between latitudes, which are longer nearer the equator and smaller nearer the poles.

P1=(φ1,λ0)=(X1,Y1,Z1) P2=(φ1,λ1)=(X2,Y2,Z2)

P4= (φ0,λ0) =(X4,Y4,Z4) P3= (φ0,λ1) =(X3,Y3,Z3)

1. Trapezoid Dimensions in latitude and longitude


The equations below define the shape of the ellipsoidal coordinates in latitude (φ) and longitude (λ), and define by an ECEF (earth-centered, earth-fixed) standard Euclidean 3D coordinate system (in meters) centered on the earth’s ellipsoidal center, where the X-Y plane passes through is the equatorial plane. The Z -axis is north (+) and south (-). The positive X-passes through the Greenwich meridian. The positive Y-passes through the 90˚ west, see Burkholder [6].

In calculating the length of a curve, we can use a numeric approximation integral. The use of "small" trapezoid (no taller than a quarter degree in latitude) will be useful for numeric integration, for the calculation of curve length (where the curve spans one upper corner and one lower corner) for a segment of the curve feature.

Similarly, for area,

1. Ellipsoidal Trapezoid (φ,λ) to ECEF (X,Y,Z)

2. Ellipsoidal Trapezoid (φ,λ) Box Area and Diagonal Lengths


The functions M(φ) is an angle (on a meridian, in radians) to distance in meters.

The function ρ(φ) is an angle (on a parallel in radians) to distances in meters.

### Using the radius of curvature as a static or varying radius of curvature

#### Using a constant radius for arc length

The radius of curvature functions are found in quite a few references, beginning with Bromford’s Geodesy and recently in IOGP, Geomatics Guidance Note 7. All equations here assume that *φ* and *λ* are expressed in radians (makes it easy to convert angles to meters), and the length functions are in meters (directly from the WGS84 ellipsoid).

A radian has arclength "r" on a circle of radius "r", the circumference is 2πr= πd.

|  |  |
| --- | --- |
| 1 radian= 57 ̊.295779513…= (180 ̊/π)1 ̊=$π$/180=0.0174532925…=1.74532925…x10-2‬Circle=2π= 6.283185307…1 radian=1/6.283185307… of the circle1 degree=π/180= .017453292519943…radian |  |

The applications work with geometry in the standard geodetic coordinate system geodetic latitude (φ), longitude (λ), and ellipsoidal height «h», if needed, (φ,λ,h). The following example deal with two corners of a latitude-longitude rectangle, with sides of two meridians and two parallels with two corners $\left(φ\_{o},λ\_{o}\right) and \left(φ\_{1},λ\_{1}\right)$ with NS and EW distances are generally less than a quarter degree.

All angle in the equation for (*φ, λ*), Δ*φ* and Δ*λ* are used in calculations in radians. All distance expressions along curves in ($φ, λ$) are in meters.

1. Length of a parallel at φ (associated to Δλ):

2. The length in meters of a segment of a parallel shall be consistent with Eq 13.

Because the parallel is a circle the radius is constant latitude, using Δλ in radians, the length in meters of an arc isso a parallel is of full length .

The distance along a meridian between  and , in radians, is the integral:

1. Length of a meridian between latitude and :

2. The length in meters of a segment of a meridian shall be consistent with Eq 14 and Eq 15.
3. Numeric approximation of length along a meridian


This is a simple programming loop:

For n=1,n; Value=0

 (Value = Value+((M(φn)+M(φn-1))×(φn- φn-1)

Length\_of\_Merridian = Value

For a line sting, the length of each "short" segment works (if the segments are short, at most a quarter degree). If a segment is longer than a quarter degree in either latitude or longitude, inserting midpoints for each "reach" 0.004363323 in radians.

### The Two Options (φ, λ) or (X, Y, Z) Integrals

The methods for geometry length and area can be calculated on the ellipsoid, or in a Cartesian 𝔼3 earth-centered, earth fixed (X, Y, Z) which is the system that is used for calculating positions in GPS. The last step is conversion from 𝔼3 (X, Y, Z) geocentric Euclidean to geographic ellipsoidal (φ,λ).

Each of the two options work is different ways. The ellipsoidal (φ,λ) equation use a trapezoid on the ellipsoid, and the calculation in the in the ECEF (X,Y,Z), see Eq 11 and Eq 10.

1. Length of a linestring in (φ,λ) and (X,Y,Z)

2. Area equivalent (φ,λ) to (X,Y,Z) using numeric integration.:

3. Length of geodetic (φ,λ) linestring to ECEF E (X,Y,Z) linestring using numeric integration:

4. Ellipsoidal Trapezoid in X,Y,Z

5. Length of a meridian from the equator and a pole.
See[34]

This sequence of numeric approximation arrives to nanometer-accuracy solution, the limits of double-precision floating point arithmetic by using a Δφ of only a quarter degree (0.0043633231 radians). Since all meridians are geometrically identical), all have a length from the equator to the pole is or approximately 10,001.965*km*. The original meter was defined as 1/10,000,000 of the distance from the pole to the equator, but the approximated meridian measurement at that time was off by 1.965km. That is impressive for a 19th century survey with a total error approximation of 0.00002%. The current meter is defined as 1/299792458 light seconds.

A quarter degree of latitude at the equator is 27.64357160*km*, but the same at the pole is 27.92349220*km*, where the polar flattening, since $b<a $as seen in the difference in *M(φ)*,the local radius of curvature, from 6,378,137*m* at the equator to 6,356,752.314245180*m* at either pole. As *M(φ)* grows smaller the length of a degree of latitude follows it.

Trapezoidal Rule for integration of a function of one variable

1. Newton's original method and a faster "Trapezoid method"


The first definition of the integral is the original from Newton (in which only one value of *f* is used in each interval usually the center of the interval). The second (Trapezoidal rule, see [58]) is a simple average Per- interval and converges in numeric integration faster than the original and works nicely in (φ,λ) because the latitude and longitudes curves are monotonic, (move in consistent manner). This numeric mechanism works especially well if *f* ismonotonic in the intervals (e.g. *f* is either constantly increasing or decreasing, which works quite well for the radius of curvature functions).

Both the "curve-feature" and "area-feature" use a version of the trapezoid method for numeric integrations.

1. The area of a polygon shall be consistent with the sum of non-overlapping areas that cover the polygon using trapezoids consistent with Eq 17.
2. The length in meters of a segment of a line string (or other curves) shall be consistent with Eq 18.

## Requirements Class: Topology Query Boolean Relations

The Ellipsoid s and Maps are both a 2-dimensional is a surface. The basic Egenhofer operators take this equivalence to define operators that are independent of distance or size such that the geometry of the "map" and the geometry of the ellipsoid are difference in areas and distance measures but are identical in "topology". The components of the topology are the boundary, ∂(geometry) and the interior, ι(geometry). The original Egenhofer relations basically match *∂∂, ιι, ∂**ι* and ι∂ which check the overlaps of boundaries and interior to simply check if they are disjoint or overlapping. This formalism and defines the operator names below so that a simple geometric system can determine if the interactions of the geometry match the formalisms. For example, the word disjoint means that the two geometries or topologies of two feature do not overlap. So "disjoint" means that the geometries of two feature do not interact. "Touch" means that the two features meet somewhere on their boundary lines. "Intersect" as for examples roads intersect if they share some positions in common (since the maps in simple features is only 2 dimensional the "intersection" can be either a crossroads or an overpass (which are identical except for elevation which "Simple Features" currently ignores).

1. Basic Topological Relations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Relation Name*** | **∂(A) Ո ∂(B)** | **ι(A) Ո ι(B)** | **∂(A) Ո ι(B)** | **ι(A) Ո ∂(B)** |
| ***Disjoint (symmetric)*** | ∅ | ∅ | ∅ | ∅ |
| ***Meets, Touches (symmetric)*** | ≠∅ | ∅ | ∅ | ∅ |
| ***Equals (symmetric)*** | ≠∅ | ≠∅ | ∅ | ∅ |
| ***Inside, Within, (reverse contains)*** | ∅ | ≠∅ | ≠∅ | ∅ |
| ***Contains (reverse inside)*** | ∅ | ≠∅ | ∅ | ≠∅ |
| ***Covers (reverse coverer-by)*** | ≠∅ | ≠∅ | ∅ | ≠∅ |
| ***Covered-by (reverse covers)*** | ≠∅ | ≠∅ | ≠∅ | ∅ |
| ***Overlaps(symmetric)*** | ∅ or ≠∅ | ≠∅ | ≠∅ | ≠∅ |

The operator "disjoint" means that the two geometries do not overlap (simple meaning of the name). The operator "meet" means the two geometries share a part of their boundaries. The operator "equal" means the two geometries are identical. The operations marked as symmetric, respond if the two sets parameters are reversed the value of the function is the same. For example, A.Overlaps(B)= B.Overlaps(A). The stated reverse operation that returns a function in the "reverse", for example A.Inside(B)=B.Contains(A).

A.Disjoint(B)=TRUE if the intersection of A and B is empty.

A.Intersects(B)=TRUE if A and B intersect.

A.Touches(B)=TRUE if A and B share a common segment of their boundaries.

A.Crosses(B) =TRUE if B spatially crosses A.

A.Within(B): =TRUE if B spatially within A.

A.Contains(B:) TRUE if B spatially contains A.

A.Overlaps(B) TRUE if B spatially overlaps A.

1. Support the topological relations by comparing if the geometry of features match the topological relations Disjoint, Intersect, Touches, Crosses, Within, Contains and Overlaps.
2. Conformance Classes

* 1. Requirements Class: Feature and Property Taxonomies (Clause 6.1)

An implementation of the Conformance class Taxonomy shall contain valid definitions for feature and property types, meeting the requirements in Clause 6.1 Requirements Class: Feature and Property Taxonomies . Each clause below shall associate the numbered requirements to a generic method of verification. Each test is associated to a single requirement defined in the requirements class.

* + 1. Req 1: Each feature or attribute type shall be accompanied by a semantically valid definition stating which real-world object might be represented or described by type.

Test to make sure that all features and attribute types have a definition consistent with the schema or taxonomy.

* + 1. Req 2: Each feature or attribute type shall be defined by a reference to a standard language-specific dictionaries.

Test to make sure that all features and attribute types all consistent with a standard language-specific dictionary.

* + 1. Req 3: Each feature instance shall be indexed by any number of semantically consistent feature types and be associated to and indexed by properties or association roles.

Test to make sure that all feature instances are indexed to consistent feature types, properties and association roles.

* + 1. Req 4: A geographic information taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role.

Test to make sure that all feature and property names are semantically valid and sufficiently complete to keep data for feature types and property types

* + 1. Req 5: The feature type component of a taxonomy shall include a "is a" type-hierarchy for features (e.g. a multiple inheritance of feature type).

Test to see if feature types in use inheritance of a subtype inherits all the properties and association types from all of its supertypes.

* + 1. Req 6: The taxonomy entry for a feature type shall contain a name, a definition, a list of any direct super-classes and any required properties.

Test to make sure that all members of a feature-type shall have a name, a definition and list of required properties which should at least contain a geometry indication its position.

* + 1. Req 7: The taxonomy entry for a property type shall contain a name, a definition, default data types able to represent the value of the property.

Check completeness of definitions of property types. The type-name can be in a common dictionary.

* + 1. Req 8: Each data entity consistent with a taxonomy shall be a dynamic object, contain a local identity associated to its name space and type names used to link it to semantically appropriate feature, property or association names and roles.

If the application can use one or both of static and dynamic object models. Check objects that are “dynamic” should be able to add dynamically attributes and associations to other features.

* + 1. Req 9: Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.

Both static and dynamic features should contain attributes or associations in the systems "dictionary" as a taxonomy.

* + 1. Req 10: Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.

Test to make sure that each feature type has a feature definition. It is sufficient that a local language dictionary contains that feature "name",

* + 1. Req 11: Every feature instance shall have a feature identifier that represents the real-world thing being described by the feature.

Every feature instance should have an object identity.

* + 1. Req 12: Any feature instance shall be consistent with the corresponding feature type and property definitions in the associated taxonomy.

Feature instances should be consistent with feature types and should be associated to properties and associations that is semantically consistent with the taxonomies or schemata in use.

* + 1. Req 13: Any feature instances that reference the same real-world phenomena shall have the same feature identifier.

If a real-world feature has two consistent instances that are consistent logically, then the instances should have the same feature-identity. This does not necessarily be its object identity which is used by the program compiler and is needed by the programing language.

* + 1. Req 14: A feature access process shall be able to collect all feature instances that refer to the same real-world entity.

Cartographic features that have more than one representational object, the applications should be able to identity all objects that are associated to the real-world feature. For example, a bridge may be part of a road feature while simultaneously be a possible obstruction for nautical travel independent of the road navigation

* + 1. Req 15: A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination.

Features can be accessed based on their feature name or type, feature location properties and any valid Boolean combinations of the full representation of the feature.

* 1. Conformance Class: 6.2 Requirements Class: Static Feature Schema
		1. Req 16: Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities.

Each property and association shall have values defined by the schemas or object definitions.

* + 1. Req 17: A feature property definition shall include a unique property name and a datatype.

Each feature property should have a unique property name and datatype.

* + 1. Req 18: Each instance of feature shall have a unique object identity.

All objects that represent features will be objects with object id's.

* + 1. Req 19: Each instance of feature shall have a feature identity.

All objects that represent features will have feature id’s.

* + 1. Req 20: Each real-world phenomenon may be referenced by multiple feature instances.
		2. Req 21: Any two features labeled with the same feature identity shall reference the same real-world phenomenon.

All features object of equal feature identities must represent the same real world feture.

* + 1. Req 22: Each feature shall have at least one feature type.

All feature objects must have at least on feature type.

* + 1. Req 23: Each feature shall be considered a member of each feature type in its type references and any of their super-types.

For operations such as query, all features having the type or any subtypes.

Any changes in properties of a feature can only change through the feature instance.

* + 1. Req 24: A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.

Properties and associations of features must be defined in an associated ontology.

* + 1. Req 25: Binary association shall be implemented as roles linking each feature in the relation to the other.

All binary relations change must update the features in all roles of the relation.

* + 1. Req 26: A feature ontology shall be fully compliant as a feature taxonomy.

Test to verify that the feature type and property hierarchical taxonomies support a “is-a" or "is a subtype of" so that object inheritance is mimicked. All feature types should be transitively a subtype of the abstract root "feature".

* + 1. Req 27: In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy, i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.

Test to verify that results from a query statement include all viable subtypes of the feature types named.

* 1. Conformance Class: Dynamic Feature Schema
		1. Req 28: A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.

Test to verify that each feature type definition is complete, including super-class and required property list. There will always be super-types or be directly inherited from the abstract root "feature".

* + 1. Req 29: A dynamic feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties and data types required by the definitions in the ontology.

Test to verify that each property type definition is complete.

* + 1. Req 30: A taxonomy shall be associated to a unique namespace. Any feature data set consistent with this taxonomy shall reference that namespace.

Test to verify that each object instance can modify its properties and feature type list within the constraints of the taxonomy.

* + 1. Req 31: A taxonomy shall contain a complete semantically valid definition for each feature type, for each property type, and for each association and role associated to the taxonomy by its namespace.

Test to verify that the taxonomy for associated properties and features are semantically consistent.

* + 1. Req 32: In any search or query, a reference to a feature-type shall include all members of the corresponding feature type and all its subtypes below it in the type-hierarchy, i.e. all such instances of those taxons which match the criteria stated in the query via inheritance.

Test to verify that search and query for feature type include all members of the type and its subtypes.

* + 1. Req 33: Any feature entity shall be associated to at least one feature definition in the feature taxonomy and any number of properties or associations consistent with its definition.

Test to verify that all feature instances of the same feature type are locatable by type, included properties and the values of properties.

* + 1. Req 35: A feature access process shall be able to collect all feature instances that refer to the same real-world entity.

Test to verify that all features that refer to the real-world entity are associated to each other. .

* + 1. Req 36: A feature access process shall be able to locate feature instances based on their feature type, the inclusion of properties, the value of those properties and spatial conditions or any valid Boolean combination of the same.

Test to verify that locate and query functions to be able to identify instances base on feature types an feature-property .

* + 1. Req 37: A feature taxonomy shall contain a non-empty set of definitions of properties of real-world phenomena that may be represented as properties associated to features in data sets consistent with this ontology.

Test to verify that properties and association roles are defined and can be associated to features. Test to verify association roles can link to features using either the feature or object identity value, or equivalent functionality.

* + 1. Req 38: Properties and association roles shall have values that are datatypes, including references to other features by either object or feature identities.

Test to make sure that all property definitions include a datatype to represent its value. This may include pointers to other features, e.g., the feature identity of a targeted feature.

* + 1. Req 39: A feature property definition shall include a unique property name and datatype.

Test by inspection that each property has a unique name, and an appropriate value consistent with its taxon definition.

* + 1. Req 40: Each instance of feature shall have a unique object identity.

Test that the feature identity of each feature is unique and can be used as a reference to implement property roles as pointers.

* + 1. Req 41: Each instance of feature shall have a feature identity.

Test that each feature has a feature identity.

* + 1. Req 42: Two features labeled with the same feature identity shall reference the same real-world phenomenon.

Test that equal feature identities always reference the same real-world entity. This does not mean that there is only one object (identity) for a feature.

* + 1. Req 43: Each feature shall have at least one feature type.

Test to ensure that feature type is mandatory for each feature instance.

* + 1. Req 44: Each feature shall be considered a member of each feature type in its type references and any of their supertypes.

Test that any retrieval of feature-by-feature type include all objects by that feature type and any of its subtypes as defined by the taxonomy hierarchy.

* + 1. Req 45: Each feature instance shall have complete ownership of all its property instances values.

Test to verify that for each feature properties the semantics are consistent with the feature type.

* + 1. Req 46: A feature property taxonomy shall contain a set of definitions of associations of real-world phenomena that may be used to express relationships between feature instances in data sets consistent with the ontology.

Test to ensure that each property instance is associated to a single feature object instance.

* + 1. Req 42: Two features labeled with the same feature identity shall reference the same real-world phenomenon.

Test to verify that all features with the same identities have the same location.

* + 1. Req 43: Each feature shall have at least one feature type.

Test to ensure that feature associations are defined by sets of association roles, e.g. by properties whose values are references to features, by feature or object identities.

* + 1. Req 44: Each feature shall be considered a member of each feature type in its type references and any of their supertypes.

Test to verify that a feature having more than one feature type are identical in position and have with compatible feature types, for example, Boulder Dam is also a highway. A dam may be associated to a hydro-electric generation.

* + 1. Req 45: Each feature instance shall have complete ownership of all its property instances values.

Test to verify that if a feature is part of another feature, then their locations are consistent. In general, the best form to associations parts of a feature are all associated to this same feature identity.

* + 1. Req 46: A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.

Test to verify that multiple features in the same position are possibly a single feature. The point may be that the same physical object can be "mapped" to the same physical object.

* + 1. Req 47: Binary association shall be implemented as symmetric roles linking each feature in the relation to the other.

Test to verify that two items in the data set that are 2 part of a relationship can be followed from each feature to the other. Binary relations must be followed from either feature to the other.

* + 1. Req 48: A feature ontology shall be fully compliant as a feature taxonomy.

Test to ensure that the ontology is associated to a taxonomy consistent with the test suite for taxonomy.

* + 1. Req 49: A feature associated to an ontology shall be fully compliant with the constraints in the associated feature ontology.

Test to verify that all constraints in the ontology must be implemented in all feature types.

* + 1. Req 50: A feature schema shall be fully compliant with a feature ontology.

Test to verify that any schema must be consistent with any feature ontology in use.

* + 1. Req 51: A feature schema’s ontology shall contain a complete object model for all associations, association roles, features, properties, and data types required by the definitions in the ontology.

Test to verify that.

* 1. Ellipsoidal Geometry Model
		1. Req 52: The length in meters of a segment of a parallel shall be consistent with Eq 13.

Test to verify that correct numeric integrations can be accurate to the meter.

* + 1. Req 53: The length in meters of a segment of a meridian shall be consistent with Eq 14 and Eq 15.

Test to verify that correct numeric integrations can be accurate to the meter.

* + 1. Req 54: The area of a polygon shall be consistent with the sum of non-overlapping areas that cover the polygon using trapezoids consistent with Eq 17.

Test to verify that correct numeric integrations can be accurate to the square meter.

* + 1. Req 55: The length in meters of a segment of a line string (or other curves) shall be consistent with Eq 18.

Test to ensure that constraints in the ontology can be tested and enforced in the feature instances. This included testing that the constraints in the ontology are not contradictory and thus preventing the creation of valid feature instances.

* 1. Topological relations
		1. Req 56: Support the topological relations by comparing if the geometry of features match the topological relations Disjoint, Intersect, Touches, Crosses, Within, Contains and Overlaps.

Test to make sure that the definitions of relations are consistent with Table 1: Basic Topological Relations.

Bibliography

Additional information and definitions were taken from the following sources as indicated, with some edits for clarity or brevity in the special context of this International Standard. Normative References are in Clause 3.

1. Andritsos, Periklis, and Keilty, Patrick. "Level-Wise Exploration of Linked and Big Data Guided by Controlled Vocabularies and Folksonomies". Advances in Classification Research 24 (2012). https:/doi.org/10.7152/acro.v24i1.14670.
2. ANSI/NISO (R2010. "Guidelines for the Construction, Format and Management of Monolingual Controlled Vocabularies". An American National Standard developed by the National Information Standards Organization, Baltimore, Maryland, USA. Approved July 5, 2006 by the American National Standards Institute, Reaffirmed May 13, 2010.
https:/www.niso.org/apps/group\_public/download.php/12591/z39-19-2005r2010.pdf.
3. Bailey, Kenneth D. Typologies and Taxonomies. An Introduction to Classification Techniques, Sage Publications International Educational and Professional Publisher. Thousand Oaks, California. 1994. ISBN 0-8039-5259-7.
4. Bishop, Richard L., and Samuel I. Goldberg. Tensor Analysis on Manifolds. Later Printing edition. New York: Dover Publications, 1980.
5. Bomford, Guy. Geodesy, Fourth edition. Oxford Science Publications: Oxford University Press, New York. 1985.
6. Burkholder, Earl F. (2017). "The 3-D Global Spatial Data Model: Principles and Applications, Second Edition". CRC Press, 2018.
7. Bunningen, Arthur H. van, Feng, Ling. Apers, Peter M. G. "Context for Ubiquitous Data Management". Proceedings of the 2005 International Workshop on Ubiquitous Data Management (UDM’05). 0-7695-2411-7/05 © 2005 IEEE.
https://www.researchgate.net/publication/221156870\_Context\_for\_Ubiquitous\_Data\_Management.
8. Butterfield, Andrew, Gerard Ekembe Ngondi, and Anne Kerr, eds. 2016. "A Dictionary of Computer Science". 7th Edition. Oxford, United Kingdom; New York, NY, United States of America: Oxford University Press.
9. Chung, TongLee, Bin Xu, Peng Zhang, Yuanhua Tan, Ping Zhu, and Adeli Wubulihasimu. "Constructing City Ontology from Expert for Smart City Management". Semantic Technology, 187–94. Lecture Notes in Computer Science. Springer, Cham, 2013. doi:10.1007/978-3-319-06826-8\_15.
10. Ciancarini, P., Poggi, F., Russo, D.: Big data quality: a roadmap for open data. In: Big Data Computing Service and Applications (Big Data Service), 2016 IEEE Second International Conference on. pp. 210–215. IEEE (2016).
11. Clapham, Christopher and James Nicholson. The Concise Oxford Concise Dictionary of Mathematics, Fifth Edition, Oxford University Press, 2014.
12. Clynch, James R. "Geodetic Coordinate Conversions," February 2006. [http: /clynchg3c.com/Technote/Tnotes.htm](http://clynchg3c.com/Technote/Tnotes.htm)
13. Clynch, James R., "[Latitude and Longitudes in Geodesy](http://clynchg3c.com/Technote/geodesy/latlons.pdf)", February 2006. [http: /clynchg3c.com/Technote/Tnotes.htm](http://clynchg3c.com/Technote/Tnotes.htm)
14. Clynch, James R., "Notes on Height Types used in Geodesy", February 2006. [http: /clynchg3c.com/Technote/Tnotes.htm](http://clynchg3c.com/Technote/Tnotes.htm).
15. Clynch, James R., "Radius of the Earth – Radii Used in Geodesy", February 2006. [http: /clynchg3c.com/Technote/Tnotes.htm](http://clynchg3c.com/Technote/Tnotes.htm).
16. Daintith, John, and R. D. Nelson, Dictionary of Mathematics, Penguin Books, London, 1989.
17. Digital Geographic Information Exchange Standard (DIGEST) Part 4 FEATURE and ATTRIBUTE CODING CATALOGUE (FACC), 2000. https:/www.dgiwg.org/digest/html/DIGEST\_2-1\_Part4.pdf
18. Donnay**,** J. D.,SphericalTrigonometry, Interscience Publishers, Inc., New York, NY, republished by Read Books Ltd., Kindle Edition.
19. Eisenhart, Luther P. An Introduction to Differential Geometry - With the Use of Tensor Calculus, Princeton University Press, republished by Read Books Ltd., Kindle Edition.
20. Egenhofer, Max J. and Andrew U. Frank, "PANDA: An Extensible DBMS Supporting Object Oriented Software Techniques" Database Systems in Office, Engineering, and Science, Zurich, Switzerland, T. Harder (ed.), Informatik Fachberichte, Vol. 204, Springer Verlag, pp. 74 79, March 1989.
21. Egenhofer, Max J. and John R. Herring, "A Mathematical Framework for the Definition of Topological Relationships", Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland, July 1990.
22. Egenhofer, Max J. and John R. Herring, "Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases", Technical Report, Department of Surveying Engineering, University of Maine, 1990 (revised versions in NCGIA Technical Report 91 7 and NCGIA Technical Report 94 1)
23. Egenhofer, M.J.; Herring, J.R. (1990). [A Mathematical Framework for the Definition of Topological Relationships](https://web.archive.org/web/20100614161335/http%3A/www.spatial.maine.edu/~max/MJEJRH-SDH1990.pdf). https://www.academia.edu/1420008/A\_mathematical\_framework\_for\_the\_definition\_of\_topological\_relationships.
24. Egenhofer, Max. Mark, David. and Herring, John. The 9-Intersection Formalism, and Its Use for Natural-Language Spatial Predicates (94-1). UC Santa Barbara, NCGIA Technical Reports, 1994.
25. Frank, Andrew U. "Tiers of Ontology and Consistency Constraints in Geographical Information Systems". International Journal of Geographical Information Science, IJGIS Special Issue (EURESCO Conference), 15, no. 7 (2001): 667–78.
26. Geodesy for the Layman. Defense Mapping Agency. https://www.ngs.noaa.gov/PUBS\_LIB/Geodesy4Layman/toc.htm.
27. Ghawi, Raji, and Nadine Cullot. "Database-to-Ontology Mapping Generation for Semantic Interoperability ". in Third International Workshop on Database Interoperability (InterDB 2007), in Conjunction with VLDB 2007, 2007.
28. GISGeography, 2015. 10 Open-Source Remote Sensing Software Packages, (Last Updated: February 18, 2018), (Accessed April 13, 2020). [https:/gisgeography.com/wgs84-world-geodetic-system/](https://gisgeography.com/wgs84-world-geodetic-system/),
29. Golden Gate Bridge. Accessed December 18, 2017. https://www.goldengate.org/bridge/history-research/.
30. Grewal, M. S., L. R. Weill and A. P. Andrews, Global Positioning Systems, Inertial Navigation, and Integration, 2007, John Wiley & Sons, Inc. https://onlinelibrary.wiley.com/doi/book/10.1002/0470099720
31. Guttman, A. (1984). "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data – SIGMOD '84. p. 47. doi: 10.1145/602259.602266
32. Harpring, Patricia. Introduction to Controlled Vocabularies: Terminology for Art, Architecture, and Other Cultural Works. Getty Publications, 2010. https://www.amazon.com/Introduction-Controlled-Vocabularies-Terminology-Architecture/dp/160606150X
33. Hedden, Heather. "The Accidental Taxonomist". 2nd edition. Information Today, Inc., 2016.
34. Herring, John R., 19-029r8 [Features and Geometries Part 2 Metrics](file:///C%3A%5CUsers%5Cgeoma%5CDocuments%5CFeatures%20and%20Geometries%20Part%202%20Metrics), 2020-04-07. https:/portal.ogc.org/files/?artifact\_id=83846&version=1.
35. Herring, John R., The Mathematical Modeling of Spatial and Non-Spatial Information in Geographic Information Systems. In D. Mark and A. Frank(Ed). Cognitive and Linguistic Aspects of Geographic Space. pp. 313-350. Kluwer Academic Publisher, Dordrecht. <https://link.springer.com/chapter/10.1007/978-94-011-2606-9_18>.
36. Hotine, Martin. Mathematical Geodesy. 1969. U.S. Department of Commerce. Environmental Science Services Administration. [https:/archive.org/details/mathematicalgeod00hoti](https://archive.org/details/mathematicalgeod00hoti).
37. IGN. "La géodésie à l'IGN". IGN, February 17, 2014. https://geodesie.ign.fr/
38. Iliffe, J., Lott, R. eds: Datums and "Map Projections: For Remote Sensing, GIS and Surveying", Second Edition. Whittles Publishing, Scotland, UK ©2008.
39. Introduction to Controlled Vocabularies. Getty Research Institute. Accessed October 19, 2017. https:/www.getty.edu/research/publications/electronic\_publications/intro\_controlled\_vocab/.
40. ISO/TC 211 Multi-Lingual Glossary of Terms. https://github.com/ISO-TC211/TMG
41. IOGP Publication 373-7-2 IOGP, Geomatics Guidance Note number 7, Part 2 – September 2019. Coordinate Conversions and Transformations including Formulas. [http: /www.espg.org](http://www.espg.org).
42. Jekeli, Christopher A. "Geometric Reference Systems in Geodesy". Division of Geodesy and Geospatial Science School of Earth Sciences, Ohio State University, July 2006. [http: /uacg.bg/filebank/att\_1855.pdf](http://uacg.bg/filebank/att_1855.pdf).
43. Jones, Andrew. "Human Geography: The Basics". Taylor and Francis. Kindle Edition. © 2012 Andrew Jones.
44. Kashyap, Vipul, Christoph Bussler, and Matthew Moran. The Semantic Web: Semantics for Data and Services on the Web. 2008 edition. Berlin: Springer, 2008.
45. Krakiwsky, E J, and D B Thomson. "Geodetic Position Computations." Geodesy and Geomatics Engineering, UNB, February 14, 1974. [http: /www2.unb.ca/gge/Pubs/LN39.pdf](http://www2.unb.ca/gge/Pubs/LN39.pdf).
46. Ligas, Marcin, and Piotr Banasik. "Conversion between Cartesian and Geodetic Coordinates on a Rotational Ellipsoid by Solving a System of Nonlinear Equations". Geodesy and Cartography 60, no. 2 (January 1, 2011): 145–59.
https:/doi.org/10.2478/v10277-012-0013-x.
47. Lipschutz, Martin M., "Differential Geometry", Schaum’s Outline Series, McGraw Hill, 1969.
48. Lund, J. D. "Topological Foundations of the Marussi-Hotine Approach to Geodesy". Geophysics Laboratory, Air Force Systems Command, United States Air Force, Hanscom Air Force Base, Massachusetts 01731-5000, October 3, 1989.
http: /www.dtic.mil/dtic/tr/fulltext/u2/a218852.pdf
49. Marussi, A. Intrinsic Geodesy. Springer Science & Business Media, 1985.
50. OECD Glossary of Statistical Terms, "Controlled Vocabulary Definition". Accessed September 19, 2017. https://stats.oecd.org/glossary/detail.asp?ID=6260.
51. OGC, [19-029r7\_Features\_and\_Geometry\_Part2\_Measure.pdf](https://portal.ogc.org/files/?artifact_id=92693). [https:/portal.ogc.org/files/?artifact\_id=92964&version=1](https://portal.ogc.org/files/?artifact_id=92964&version=1).
52. Oxford University Press, A Dictionary of Computer Science, 7th Edition 2016
https://global.oup.com/academic/product/a-dictionary-of-computer-science-9780199688975?cc=us&lang=en&.
53. Panigrahi, Narayan (2014). Computing in Geographic Information Systems. 1st edition. Boca Raton: CRC Press.
54. Panou, Georgios, and Romylos Korakitis. "Geodesic Equations and Their Numerical Solutions in Geodetic and Cartesian Coordinates on an Oblate Spheroid." Journal of Geodetic Science 4, no. 2017; 7 (May 2017): 31–42.
https:/doi.org/10.1515/jogs-2017-0004.
55. Pearson II, Frederick (1990). Map Projections: Theory and Applications. CRC Press
56. Snyder, John P. "Map Projections: A Working Manual". USGS Numbered Series. Professional Paper 1395. Washington, D.C.: U.S. Government Printing Office, 1987. http: /pubs.er.usgs.gov/publication/pp1395.
57. Spivak, Michael. A Comprehensive Introduction to Differential Geometry. Vol 1-5. Publish or Perish, Inc., 1999.
58. Spivak, Michael. Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus. 5th edition. Cambridge, Mass: CRC Press, 1971. http: /strangebeautiful.com/other-texts/spivak-calc-manifolds.pdf
59. Stolze, Knut. (2003). SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System. January 2003
60. Struik, Dirk J. Lectures on Classical Differential Geometry: Second Edition (Dover Books on Mathematics). Dover Publications. 1961.
61. Techopedia. Dynamic Database Management System, explains Dynamic Database Management System (Dynamic DBMS)
https://www.techopedia.com/definition/19044/dynamic-database-management-system-dynamic-dbms
62. Techopedia Object Model. https://www.techopedia.com/definition/8635/object-model.
63. Teng, Wei-Kuo, et. al. A Comparison of Great Circle, Great Ellipse and Geodesic Sailing, Journal of Marine Science and Technology, Vol. 21, No. 3, pp. 287-299. 2013. http: /jmst.ntou.edu.tw/marine/21-3/287-299.pdf

1. [The Common Lisp Foundation](http://cl-foundation.org/). "Welcome to Common-Lisp.net!" Accessed August 9, 2017. [https:/common-lisp.net/](https://common-lisp.net/).
2. Thomas, C. M. and W. E. Featherstone (February 2005), "Validation of Vincenty's Formulas for the Geodesic Using a New Fourth-Order Extension of Kivioja's Formula", Journal of Surveying Engineering, Vol. 131, No. 1. https:/espace.curtin.edu.au/bitstream/handle/20.500.11937/36501/134983\_134983.pdf?sequence=2
3. Topological and Thematic Layering of Geological Map Information. Accessed October 27, 2017.
https:/pubs.usgs.gov/of/1997/of97-269/viljoen.html.
4. Torge, Wolfgang. Geodesy. 3rd edition. Berlin; New York: De Gruyter, 2001.
[http: /fgg-web.fgg.uni-lj.si/~/mkuhar/Zalozba/Torge-Geodesy(2001).pdf](http://fgg-web.fgg.uni-lj.si/~/mkuhar/Zalozba/Torge-Geodesy%282001%29.pdf)
5. Uschold, Michael. "Ontologies and Database Schema: What’s the Difference?" Accessed August 9, 2017.
https:/pdfs.semanticscholar.org/b44f/a4592b69183c1965d0075dea1a3bc58dfbfe.pdf
6. Vincenty, T., "Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations". Survey Review 23, no. 176 (April 1975). https:/www.ngs.noaa.gov/PUBS\_LIB/inverse.pdf
7. Wikipedia, DE-9IM. <https://en.wikipedia.org/wiki/DE-9IM>.
8. Wikipedia, Human geography. https://en.wikipedia.org/wiki/Human\_geography
9. Wikipedia, Geodesics on an ellipsoid, <https://en.wikipedia.org/wiki/Geodesics_on_an_ellipsoid>
10. Wikipedia, Geodesic curvature, https://en.wikipedia.org/wiki/Geodesic\_curvature
11. Wikipedia, Geoid. [https:/en.wikipedia.org/wiki/Geoid](https://en.wikipedia.org/wiki/Geoid)
12. Wikipedia, Global Positioning System. [https:/en.wikipedia.org/wiki/Global\_Positioning\_System](https://en.wikipedia.org/wiki/Global_Positioning_System)
13. Wikipedia, History of the metre, [https:/en.wikipedia.org/wiki/History\_of\_the\_metre](https://en.wikipedia.org/wiki/History_of_the_metre).
14. Wikipedia, Monotonic function. [https:/en.wikipedia.org/wiki/Monotonic\_function](https://en.wikipedia.org/wiki/Monotonic_function)
15. Wikipedia, Numerical integration. <https://en.wikipedia.org/wiki/Numerical_integration>
16. Wikipedia, Separation of concerns. https://en.wikipedia.org/wiki/Separation\_of\_concerns
17. Wikipedia, Trapezoidal rule. [https:/en.wikipedia.org/wiki/Trapezoidal rule](https://en.wikipedia.org/wiki/Trapezoidal_rule).
18. WGS84. http://wiki.gis.com/wiki/index.php/WGS84.
19. Zund, Joseph. Foundations of Differential Geodesy. 10.1007/978-3-642-79187–1. Springer Verlag, Berlin, 1994.
1. This implies that constraints in a formal ontology should be held submissive to reality, so that a single valid counterexample for a constraint, should imply that the ontology is in question, not the reality. [↑](#footnote-ref-2)
2. In the case of this specification, the domain knowledge is mainly in the taxonomy and ontology associate to the data. In geographic information the domain is the "feature representing a real-world phenomenon". [↑](#footnote-ref-3)
3. The GIS definition used "real-world phenomena" and Simula used "real world objects". This is an extension needed for GIS since many "features" displayed in data sets are "conceptual" like political boundaries, buffer zones, voting districts and others existing in the real world has real world implications, but is not always a real-world physical but a conceptual "object". [↑](#footnote-ref-4)
4. The [USSD](https://www.ussdams.org/) (US Society of Dams, https://www.ussdams.org/) recognizes [12 basic types of dams](https://www.ussdams.org/dam-levee-education/overview/types-of-dams/); a weir is an "overflow dam". [↑](#footnote-ref-5)
5. Euphonious and alliterative jargon for the "entire useful electromagnetic spectrum" ordered by frequency, direct current (0 Hz) to light (400-770 THz) and now maybe up to gamma rays (30EHz). [↑](#footnote-ref-6)