OGC® DOCUMENT: 08-131R4

OGC®
Making location count.

THE MODSPEC MODEL - PART 1:CORE - A STANDARD FOR DESIGNING AND WRITING MODULAR STANDARDS

DRAFT STANDARD

DRAFT

Version: 1.1.0
Submission Date: 2025-03-18
Approval Date: 2025-xx-xx
Publication Date: 2025-xx-xx
Editor: Carl Reed, John Herring, Chuck Heazel

License Agreement

Use of this document is subject to the license agreement at https://www.ogc.org/license

Copyright notice

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I PREFACE

This OGC member developed and approved document, referred to as the `ModSpec`, defines a model and related requirements and recommendations for writing and structuring modular standards documents. Further, this model is designed to enable consistent and verifiable testing of implementations of a standard that claim conformance. The ModSpec is a meta-standard: A standard specifying requirements for crafting and documenting modular and testable standards.

The goals of using the ModSpec are:

- To define characteristics and a structure for the development of modular standards which will minimize the difficulty in writing testable standards while maximizing usability and interoperability.
- To ensure that a standard specifies requirements in a common and consistent manner and that these requirements are testable.

NOTE: Historically, this document has been known and abbreviated as the “ModSpec”. For continuity and ease of understanding this document may also be referred to as the “OGC ModSpec”.

Suggested additions, changes, and comments on this document are welcome and encouraged. Such suggestions may be submitted by creating an issue in the OGC ModSpec GitHub repository (https://github.com/opengeospatial/ogc-modspec).

II SECURITY CONSIDERATIONS

No security considerations have been made for this document.

III SUBMITTING ORGANIZATIONS

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

- Carl Reed, Charles Heazel, ImageMatters

IV DOCUMENT TERMS AND DEFINITIONS

This document uses the standard terms defined in Subclause 5.3 of the OGC Web Services Common Standard [OGC 06-121r9], which is based on the ISO/IEC Directives, Part 2: 2021. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the imperative verb form used to indicate a requirement to be strictly followed to conform to the ModSpec.

V DOCUMENT EDITORS

The following OGC Members participated in editing this document:

PERSON	ORGANIZATION REPRESENTED
Carl Reed	Carl Reed & Associates
Chuck Heazel	Charles Heazel

VI DOCUMENT CONTRIBUTORS

The following OGC Members contributed and particpated in developing Version 1.1 of the ModSpec.

PERSON	ORGANIZATION REPRESENTED
Carl Reed	Carl Reed
Chuck Heazel	Charles Heazel
Jeff Yutzler	ImageMatters

VII REVISION HISTORY

This is the second normative version of this document.

VIII FUTURE WORK

This version of the ModSpec restructures the document into a multi-part standard. This document is Part 1 — the core. It provides a technology agnostic model for modular standards. Future "parts" will be specific for different technologies. Planned extensions include:

- ModSpec Part providing requirements and recommendations for specifying requirements and conformance tests using RDFS, SHACL, and OWL.
- ModSpec Part providing requirements and recommendations for specifying requirements and conformance tests using JSON.

In addition, improvements to this document will be made based on implementation and changing technical requirements.

IX FOREWORD

The OGC ModSpec — A Standard for Designing and Writing Modular Standards specifies a formal structure and requirements for writing modular standards documents. However, the ModSpec does not supply specific content. Where possible, this document is conformant with itself (with respect to the core requirements class and the conformance test class, Clause 8 and the Conformance Test Suite Annex A.1).

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 Inc. shall not be held responsible for identifying any or all such patent rights.

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

1

SCOPE

1 SCOPE

This OGC Standard for Designing and Writing Modular Standards, also known as the `ModSpec`:

- Specifies rules for the internal structure and organization of a standard.
- Defines requirements for specifying the structure of a standards document as organized sets of criteria, those that are to be tested ("requirements") and those that are not tested ("recommendations" and "permissions").
- Is designed to enable the clear and concise specification of requirements (the *shalls* or *musts* in a standard) that fully supports the ability to define implementable conformance tests.
- Formalizes implementing the requirements specified in the ModSpec so that reusable, modular standards can be developed.

The standardization goal of the ModSpec is to define characteristics and a structure for the specification of modular and testable Standards that will encourage implementation by minimizing difficulty determining requirements, mimicking implementation structure, and maximizing usability and interoperability. The ultimate goal of this approach is to enable interoperable implementations of a standard to be tested and deemed *conformant* or not.

The one standardization target type for the ModSpec is modular standards. While the primary focus is on modular OGC standards, this approach should prove applicable to standards for any domain.

1.1. Understanding the ModSpec

Reading the Terms and Definitions clause will help in understanding the content and requirements stated in this document.

The Standards Fundamentals clause provides a more detailed introduction to the fundamental concepts used in the creation of this document.

Also read Annex C. Annex C defines the UML model upon which the ModSpec is based. Annex C also contains informal and non-normative definitions ordered for ease of understanding. These two sections can be read first to aid in the understanding of the rest of the document.

NOTE: Please note that the ModSpec has been approved by the OGC Membership as a policy directive for the development and revision of any OGC Standard or Abstract Specification that has requirements. However, the ModSpec is written to be non-OGC specific and can be used by any Standards Development Organization (SDO) as a formal guide for structuring a standards document.

1.2. ModSpec document structure

Version 2.0 of the ModSpec is split into a Core standard and multiple Parts. These are:

- Core: contains all the core requirements and informational text that define the model and internal structure of a standard.
- Part 1: UML Model requirements
- Part 2: XML and Schematron Model requirements

Future Parts to the ModSpec Standard may include:

- Part 3: RDF/OWL requirements
- Part 4: JSON Schema

2

CONFORMANCE

2 CONFORMANCE

Conformance to the ModSpec by a modular standard can be tested by inspection. The test suite is in Annex A.

Part 1 of the ModSpec (this document) has one requirements class and one related conformance class:

- The Core: Common requirements for specifying standards documents. See Clause 8 and Annex A.1

The ModSpec contains normative language and thus places requirements on conformance, or mechanism for adoption, of candidate standards to which the ModSpec applies. In particular:

- ModSpec Requirements Class: Core specifies the core requirements which shall be met by all standards claiming conformance to the ModSpec.
- Mapping the ModSpec to types of models gives information on how the ModSpec is to be applied to extensions to the core model for requirements and conformance clauses.

Such extensions are defined in additional Parts (volumes) to the ModSpec Standard.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC: ISO/IEC 10000-1:1998, *Information technology — Framework and taxonomy of International Standardized Profiles Part 1: General principles and documentation framework*. ISO, IEC (1998).

ISO/IEC DIR 2, *ISO/IEC Directives, Part 2*. https://www.iso.org/sites/directives/current/part2/index.xhtml.

ISO: ISO 19105:2022, *Geographic information — Conformance and testing*. International Organization for Standardization, Geneva (2022). https://www.iso.org/standard/76457.html.

OMG Unified Modeling Language (OMG UML), Infrastructure, V2.5, OMG Document Number: formal/2015-03-01, Standard document URL: https://www.omg.org/spec/UML/2.5

OMG Unified Modeling Language (OMG UML), Superstructure, V2.4.1, OMG Document Number: formal/2012-05-07; Standard document URL: https://www.omg.org/spec/UML/ISO/19505-2/PDF

5

TERMS AND DEFINITIONS BY CATEGORY

5 TERMS AND DEFINITIONS BY CATEGORY

For the purposes of this document, the following terms and definitions shall apply. Terms not defined here take their meaning from computer science or from their Standard English (common US and UK) meanings. The form of the definitions is defined by ISO Directives.

Many of these definitions depend upon the model given in Annex C: ModSpec UML Model, Semantics, and Definitions.

5.1. Building Blocks Terms and Definitions

5.1.1. Building Block

a requirements class or a requirements module with no direct dependencies on other requirements classes or modules and their associated compliance test class or compliance test module.

5.1.2. Dependency

a module (the target) which is used, produced or associated to by an implementation of the source module.

5.1.3. Direct Dependency

another requirements class (the dependency) whose requirements are defined to also be requirements of this requirements class

NOTE: A direct dependency (of a requirements class) of the current requirements class will have the same standardization target as the current requirements class. This is another way of saying that the current requirements class extends, or uses all the aspects of the direct dependency (or a requirements class). Any tests associated to this dependency can be applied to this requirements class.

When testing a direct dependency of a requirements class, the standardization target is directly subject to the test in the specified conformance test class of the direct dependency of a requirements class.

5.1.4. Extension

requirements class which has a direct dependency on another requirements class

NOTE: Here an extension of a requirements class is defined on requirements class so that their implementation may be software extensions in a manner analogous to the extension relation between the requirements classes.

5.1.5. Indirect Dependency

requirements class with a different standardization target which is used, produced or associated to by the implementation of this requirements class

NOTE: In this instance, as opposed to the direct dependency of a requirements class, the test against the consumable or product used or produced by the requirements class does not directly test the requirements class, but tests only its side effects. Hence, a particular type of feature service could be required to produce valid XML documents, but the test of validity for the XML document is not directly testing the service, but only indirectly testing the validity of its output. Direct dependencies test the same standardization target, but indirect dependencies test related but different standardization targets.
For example, if a DRM-enabled service is required to have an association to a licensing service, then the requirements of a licensing service are indirect requirements for the DRM-enabled service. Such a requirement may be stated as the associated licensing service has a certificate of conformance of a particular kind.

5.1.6. Leaf Package

UML model package that does not contain any subpackages, but contains classifiers

5.1.7. Module

each of a set of standardized parts or independent units that can be used to construct a more complex structure

Note: The concept ‘module’ is key to the ModSpec structure and model. Modules have a direct relationship to the concept of building blocks. The power of the concept is that modules can be reused in different systems (specify once, use many). Modules can be imported from other programs. Modules can be replaced without affecting the rest of the system.

5.1.8. Profile

specification or standard consisting of a set of references to one or more base standards and/or other profiles, and the identification of any chosen conformance test classes, conforming subsets, options and parameters of those base standards, or profiles necessary to accomplish a particular function.

NOTE: In the usage of the ModSpec, a profile will be a set of requirements classes or conformance classes (either preexisting or locally defined) of the base standards.

This means that a standardization target being conformant to a profile implies that the same target is conformant to the standards referenced in the profile.

5.2. Document Terms and Definitions

5.2.1. Best Practice

a specification that has been approved by a legitimate Standards Body as a recommendation for implementation.

5.2.2. Certificate Of Conformance

evidence of conformance to all or part of a standard, awarded for passing one or more of the conformance test classes specified in that standard

NOTE: Certificates‖ do not have to be instantiated documents; having proof of passing the conformance test class is sufficient. For example, the OGC currently keeps an online list of conformant applications at http://www.opengeospatial.org/resource/products. Each certificate of conformance is awarded to a standardization target.

5.2.3. Informative Statement

expression in a document conveying non-normative information

NOTE: Includes all statements in a document not part of the normative requirements, recommendations, permissions, or conformance tests. Included for completeness.

5.2.4. Normative Statement

expression in a document conveying information required to define conformance

NOTE: Includes all normative statements in a document including requirements, recommendations, permissions, and conformance tests. Included for completeness.

5.2.5. Specification

document containing recommendations, requirements, permissions, and conformance tests

NOTE 1: This definition is included for completeness.

NOTE 2: In the OGC, there are Abstract Specifications and Implementation Standards. Abstract Specifications may of may not be testable. Further, Abstract Specifications may not be directly implementable. Implementation Standards are always testable and contain a conformance test suite.

5.2.6. Standard

a specification that has been approved by a legitimate Standards Body

NOTE 1: This definition is included for completeness. Standard and specification can apply to the same document. While specification is always valid, standard only applies after the adoption of the document by a legitimate standards organization.

NOTE 2: A standard should consist primarily of Normative Statements. The goal should be for the standard to be concise. Supporting information can be provided through a users guide.

5.2.7. Statement

expression in a document conveying information

5.2.8. Users Guide

Non-normative information that assists in understanding a standard but is not required to implement the standard.

5.3. Core Terms and Definitions

5.3.1. Conformance Class

a set of conformance tests that must be passed to receive a single certificate of conformance

NOTE: When no ambiguity is possible, the word test may be left out, so conformance test class may be called a conformance class.
In the ModSpec, the set of requirements tested by the conformance tests within a conformance class is a requirements class and its dependencies. Optional requirements will be in a separate requirements class with other requirements that are part of the same option. Each requirements class corresponds to a separate conformance class
Each requirements class will be in a 1 to 1 correspondence to a similarly named conformance class that tests all of the requirements in the requirements class.
All requirements in a conformance class will have the same standardization target type.

5.3.2. ConformanceModule

a set of related conformance classes and their associated components.

5.3.3. Conformance Suite

set of conformance modules that define tests for all requirements of a standard or abstract specification

NOTE: The **conformance suite** is the union of all **conformance modules** and their associated conformance classes. It is by definition the **conformance class** of the entire **standard** or **abstract specification**.

5.3.4. Conformance Test

test, abstract or real, of one or more requirements contained within a standard, or set of standards

5.3.5. Conformance Test Method

How an implementation of the standard is tested for conformance. Testing may be automated, manual, or a hybrid.

NOTE: Testing by an independent second party is recommended.

5.3.6. Core Requirements Class

unique requirements class that must be satisfied by any conformant standardization targets associated to the standard

NOTE: The core requirements class is unique because if it was possible to have more than one, then each core would have to be implemented to pass any conformance test class, and thus would have to be contained in any other core. The core may be empty, or all or part of another standard or related set of standards.

The core can refer to this requirements class, its associated conformance test class or the software module that implements that requirements class.

5.3.7. Model

A representation of those aspects of the standardization target type which are to be governed by a standard. The model captures both the conceptual and logical properties of the

standardization target type. The requirements promulgated by a standard should be expressed in terms of those conceptual and logical properties.

In short, the model provides the vocabulary for expressing requirements.

5.3.8. Permission

uses “may” and is used to prevent a requirement from being “over interpreted” and as such is considered to be more of a “statement of fact” than a “normative” condition.

5.3.9. Recommendation

expression in the content of a standard conveying that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited

NOTE 1: Although using normative language, a recommendation is not a requirement. The usual form replaces the shall (imperative or command) of a requirement with a should (suggestive or conditional).

NOTE 2: Recommendations are not tested and therefore have no related conformance test.

5.3.10. Requirement

expression in the content of a standard conveying criteria to be fulfilled if compliance with the standard is to be claimed and from which no deviation is permitted

NOTE: Each requirement is a normative criterion for a single type of standardization target. In the ModSpec, requirements are associated to conformance tests that can be used to prove compliance to the underlying criteria by the standardization target. The implementation of a requirement is dependent on the type of standard being written. A data standard requires data structures, but a procedural standard requires software implementations. The view of a standard in terms of a set of testable requirements allows us to use set descriptions of both the standard and its implementations. The specification of a requirement is usually expressed in terms of a model of the standardization target type, such as a UML model, or an XML or SQL schema. Anything without a defined test is a-priori not testable and thus would be better expressed as a recommendation. Requirements use normative language and in particular are commands and use the imperative “shall” or similar imperative constructs. Statements in standards which are not requirements and need to be either conditional or future tense normally use “will” and should not be confused with requirements that use “shall” imperatively

5.3.11. Requirements Class

an aggregate of requirements with a single standardization target type that must all be satisfied to pass a conformance test Class.

NOTE: There is some confusion possible here, since the testing of indirect dependencies seems to violate this definition. But the existence of an indirect dependency implies that the test is actually a test of the existence of the relationship from the original target to something that has a property (satisfies a condition or requirement from another requirements class).

5.3.12. Requirements Module

a set of related requirement classes and their associated components.

5.3.13. Standardization Goal

a concise statement of the problem that the standard helps address and the strategy envisioned for achieving a solution. This strategy typically identifies real-world entities that need to be modified or constrained. At the abstract level, those entities are the Standardization Target Types.

5.3.14. Standardization Target

entity to which some requirements of a standard apply

NOTE: The standardization target is the entity which may receive a certificate of conformance for a requirements class.

5.3.15. Standardization Target Type

type of entity or set of entities to which the requirement of a standard apply

NOTE: For example, the standardization target type for The OGC API – Features Standard are Web APIs. The standardization target type for the CDB Standard is “datastore”. It is important to understand that a standard’s root standardization target type can have sub-types, and that there can be a hierarchy of target types. For example, a Web API can have sub types of client, server, security, and so forth. As such, each requirements class can have a standardization target type that is a sub-type of the root.

5.3.16. will

In informative sections, the word “will” implies that something is an implication of a requirement. The “will” statements are not requirements, but explain the consequence of requirements.

6

CONVENTIONS

6 CONVENTIONS

6.1. Symbols (and abbreviated terms)

All symbols used in this document are either:

1. Common mathematical symbols
2. UML 2 (Unified Modeling Language) as defined by OMG and accepted as a publicly available standard (PAS) by ISO in its earlier 1.3 version.

6.2. Identifiers

The normative provisions in this standard are denoted by the URI namespace

https://www.opengis.net/spec/modspec-1/1.1/

All requirements that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of “req” in a requirement URI denotes:

https://www.opengis.net/spec/modspec-1/1.1/

An example might be:

/req/core/crs

All conformance tests that appear in this document are denoted by partial URIs which are relative to the namespace shown above.

For the sake of brevity, the use of “conf” in a requirement URI denotes:

https://www.opengis.net/spec/modspec-1/1.1/

The same convention is used for permissions (per) and recommendations (rec).

6.3. Abbreviated terms

In this document the following abbreviations and acronyms are used or introduced:

ERA	Entity, Relation, Attribute (pre-object modeling technique)
ISO	International Organization for Standardization (from Greek for “same”)
OGC	Open Geospatial Consortium (http://www.opengeospatial.org/)
OMG	Object Management Group (http://www.omg.org/)
SQL	ISO/IEC 9075 query language for relational databases, not originally an acronym, but now often cited as “Structured Query Language”
TC	Technical Committee (usually either in ISO or OGC)
UML	Unified Modeling Language (an object modeling language)
XML	eXtensible Markup Language

6.4. Finding requirements and recommendations

Each normative statement in the ModSpec is stated in one and only one place, in a standard format, with an unique label, such as REQ001, REC001, or PER001. A requirement, recommendation, or permission may be repeated for clarification. The statement with the unique label is considered the official statement of the normative requirement or recommendation.

In this document, all requirements are associated with tests specified in the test suite in Annex A. The reference to the requirement in the test case is done by a requirements label. Recommendations and permissions are not tested although they still are documented using a standard template and have unique identifiers.

Requirements classes are separated into their own clauses and named, and specified according to inheritance (direct dependencies). The Conformance test classes in the test suite are similarly named to establish an explicit and link between requirements classes and conformance test classes.

7

STANDARDS FUNDAMENTALS

7 STANDARDS FUNDAMENTALS

7.1. Building Blocks

In software development technology, there is a concept called *building block*. In software development, building blocks are used to support the software build process where source code files/libraries can be accessed from multiple sources, converted into executable code, and linked together in the proper order until a complete set of executable files is generated. The same concept can be applied to OGC Standards development: Requirements classes can be linked together from one or more standards to create a new standard not originally envisioned when the requirements were originally defined.

The Open Group suggests that building blocks have the following characteristics:

1. A building block is a package of functionality defined to meet business or domain needs.
2. A building block may interoperate with other, inter-dependent, building blocks.
3. A good building block has the following characteristics:
    a) Considers implementation and usage, and evolves to exploit technology and standards.
    b) May be assembled from other building blocks.
    c) May be a subassembly of other building blocks.
    d) Ideally a building block is re-usable and replaceable, and well specified.
4. A building block may have multiple implementations but with different inter-dependent building blocks.

These characteristics are slightly modified from the Open Group definitions to accommodate the use of the building block concept in standards work.

7.2. Standardization Context — Goals and Targets

Every standards document should include a Standardization Goal. This is a concise statement of the problem that the standard helps address and the strategy envisioned for achieving a solution. This strategy typically identifies real-world entities that need to be modified or

constrained. At the abstract level, those entities are the Standardization Target Types. These are the classes of entities to be standardized. A Standard defines the requirements levied on one or more Standardization Target Types.

Instances of a Standardization Target Type are the Standardization Targets. These are the real-world manifestations of the Standardization Target Type. In summary:

- Standardization Goal – identifies the problem and identifies the actors and entities involved in solving that problem
- Standardization Target Type – An abstract representation of one of the actors or entities identified in the Standardization Goal
- Standardization Target – an implementation of a Standardization Target Type. These are the real-world entities which can be tested for conformance with the requirements documented in the Standard.

Standardization Target Types can be hierarchical. The Conceptual, Logical, Physical hierarchy is one example where the Standardization Target Types are information models. Another example would be implementations of OGC API — Features Part 2 that support XML data exchange.

The Standardization Target Types, Standardization Targets, and Standardization Goal provide a well-defined context for the standard. This will help users of standards to quickly understand the scope of a standard and to select those standards appropriate for their needs. It also will help keep standards developers focused on the intended use of their standards, avoiding standards which are overly broad and/or unfocused.

7.3. Conformance, Requirements, and key information

In the conformance test suite, there will be a test defined to verify the validity of the claim that an implementation of the standard (standardization target) satisfies each mandatory requirement specified in the standard. Since the normative language of the body of the standard and the conformance test classes both define what conformance to the standard means, they will be equivalent in a well-written standard. The ModSpec requires a standards document to be well-written, at least in stating requirements and conformance tests.

Conformance tests are aggregated into conformance classes that specify how certain “certificates of conformance” are achieved. The natural inclination is to aggregate the requirements. The issue that blocks this approach is that some requirements are optional while others are mandatory. To achieve a cleaner separation of requirements, the ModSpec separates them into sets (called “requirements classes”), each of which has no optional components. Since the normative statement of each requirement is only declared once and is uniquely identified as such, each requirement will be in a clause associated to its requirements class.

Therefore, the ModSpec defines a “requirements class” as a set of requirements that must all be passed to achieve a particular conformance class (see Clause 5.3.1). Each requirements class

has a one-to-one correspondence with a conformance class. A standard written to the ModSpec may use this "module" structure in any manner consistent with the rest of the ModSpec.

A standard may have mandatory and optional requirements classes. This allows the options in the testing procedure to be grouped into non-varying mandatory and optional conformance classes. Each requirement within an optional requirements class is mandatory when that requirements class is implemented. When needed, a particular requirements class may contain only a single requirement.

However, care must be taken, since the requirements classes may not always in a one-to-one correspondence to conformance classes in other standards which may be the source of requirements for a standard conformant to the ModSpec. If other standards are used, their options shall be specified to be useable within a standard conformant to the ModSpec, see Clause 8.1.4.1.

Conformance classes may contain dependencies on one another. These are represented by tests in one conformance class that state that another conformance class must be passed to qualify to pass this conformance class. In terms of requirements, that says that the dependent conformance class contains tests (by reference) for all requirements of the "included" conformance class.

As defined in the ModSpec, one requirements class is dependent on another if the other is included through such a reference. In this manner, requirements classes can be treated as sets of requirements (each in a single requirements class but included in others by reference to its "home" requirements class).

In the ModSpec, each conformance requirement is separated in its own labeled paragraph, such as seen in Requirement 1 below.

The distribution of the information in a standard is not restricted. The only requirement is that requirements be grouped in a manner consistent with the conformance test classes, see Requirement 6 and Requirement 7.

7.4. Documenting the Standard

NOTE: OGC Standards are written using an OGC Member approved template that is conformant with the requirements stated in the ModSpec

This template should be specified by the following descriptions:

1. A standards document contains Clauses (corresponding to numbered sections as they might appear in a table of contents) which describe its standardization target type and its requirements.

2. A standard contains Annexes or is associated to other documents (both a logical type of Clause), one of which is the Conformance Test Suite (which may be an abstract description of the test suites to be implemented separately). In OGC Documents, this is Annex A – Abstract Test Suite.

3. All requirements, recommendations, permissions, and models are introduced and defined first in the numbered Clauses.

4. All requirements are identifiable as requirements.

5. All requirements in a document are uniquely numbered.

6. All tests for conformance to those requirements are defined in the Conformance Test Suite.

7. Tests are be grouped for convenience into conformance test classes and if desired the classes are grouped into conformance test modules.

8. The tests, if conducted, determine to some degree of certainty whether an implementation meets the requirements which the tests reference.

9. The tests are organized into some number of conformance "classes" where each conformance class has a one to one relationship with a requirements class. If a standard does not do this, it is has by default only one "conformance class".

10. Certificates of conformance are awarded by a testing entity based on these conformance classes.

11. There is a clear distinction between normative and informative parts of the text.

12. Examples and notes are informative, and do not use "normative" language.

A UML representation of important properties of this model is given in Annex C, Section 2.

8

MODSPEC REQUIREMENTS CLASS: CORE

8 MODSPEC REQUIREMENTS CLASS: CORE

This clause specifies the requirements, recommendations, and permissions for the content and structure of a modular standard. This collection of requirements, recommendations, and permissions are also known as the `core` of the ModSpec. All the requirements specified in this ModSpec core comprise the ModSpec Core Requirements Class.

8.1. General Requirements

The following requirement states that every requirement in a standards document is associated with one and only one requirements class(es).

Table 1

Requirement 0	**/req/core/reqs-are-in-class**
A	Each requirement in a standard *SHALL* be associated with a requirements class.

There may be one or more requirements in a requirements class.

The following requirement states that every requirement is testable.

REQUIREMENT 1	
IDENTIFIER	`/req/core/reqs-are-testable`
A	All the parts of a requirement *SHALL* be testable.
B	Failure to pass any part of a requirement *SHALL* be a failure to pass the associated conformance test.

NOTE 1: This further means that failure to pass the test specified for a part of requirement is a failure to pass the requirement. Therefore, by definition, any requirements class that contains the requirement that failed to pass also fails to pass.

Therefore, by definition, any requirements class that contains the requirement that failed to pass also fails to pass.

The following requirement states that every component of a standard will have a unique identifier

REQUIREMENT 2	
IDENTIFIER	`/req/core/all-components-assigned-uri`
A	Each component of a standard, including requirements, requirements modules, requirements classes, conformance test, conformance modules, and conformance test classes *SHALL* be assigned a unique identifer or label.

NOTE 2: In the OGC, the enforcement of this requirement and its associated recommendation is the purview of the OGC Naming Authority or its equivalents.

The following recommendation encourages the consistent use of these unique identifiers/labels in the standard as well as any external documentation referencing that standard.

RECOMMENDATION 1	
IDENTIFIER	`/rec/core/uri-external-use`
A	These unique identifiers/labels *SHOULD* be used in any external documentation that references component elements of a standard in a normative manner, including but not limited to other standards, implementations of the conformance test suite, or certificates of conformance for implementations conformant to the standard in question.

While a requirement may be referenced in more than one place in a standard, the normative definition of a requirement shall be its "**home**" (see Clause 6.4) and will be the only place where full normative language is used.

The following permissions relate to possible content specified in the core of a standard.

Table 2

PER001	**/per/core/informational-content-in-core** The informational and structural universals of the standard *MAY* be included in the core text of the standard without violations of the ModSpec. This is true if the requirements of the extension are not implicit in what is included in the core.

In this manner, the core requirements class and its associated content can be thought of not only as the requirements of the core conformance class, but as a form of reference model for establishing core vocabularies and schemas for the entire standard.

Table 3

PER002	**/per/core/core-may-contain-schema-terms** The core *MAY* contain the definition and schema of commonly used terms and data structures for use in other components and/or structures throughout the standard.

The following states how and where vocabularies are specified in relation to a requirements class.

REQUIREMENT 3	
IDENTIFIER	`/req/core/vocabulary-and-parent-req-class`
A	Requirements on the use and interpretation of vocabulary *SHALL* be in the requirements class where that use or interpretation is used.

Table 4

PER004	**/per/core/external-vocabs-core** Importation of external vocabularies and schemas *MAY* be in the core of a standard.

Example: In the specification of a metadata service, the Dublin Core concept of a “Title” and the XML schema structure used for its specification can be in the core of the service specification. How a particular request-response pair uses the data structure to mean the title of a particular document or dataset will be specified in the requirements class in which the request-response pair is defined and set against requirements.

8.1.1. Using the model

The primary difficulty in speaking about standards (or candidate standards) as a group is their diverse nature. Some standards use UML to define behavior, others use XML to define data structures, and others use no specific modeling language at all. However, they all must model the standardization target type to which they apply since they need to use unambiguous language to specify requirements. Thus, the only thing they have in common is that they define testable requirements against some model of an implementation of the standard (the standardization target type). For completeness, they should also specify the conformance tests for these requirements that are to be run for validation of the implementations against those requirements.

The assumption is that each standard has a single (root) standardization target type from which all extensions inherit. If this is not true, then the standard can be logically factored into parts each corresponding to a “root” standardization target type, and that the standard addresses each such part separately (see the definition of requirements class). In this sense, the next requirement divides the standard into parts more than restricting their content.

REQUIREMENT 4	
IDENTIFIER	`/req/core/single-standardization-target-type`

REQUIREMENT 4	
A	Each requirement class in a conformant standard *SHALL* have a single standardization target type.

In practice, the standardization target type of the core requirements class is the root of an inheritance tree where extensions all have the core's target type as an ancestor, and thus can be considered as belonging to the same "class" or type as the core's target type.

REQUIREMENT 5	
IDENTIFIER	`/req/core/test-class-single-standardization-target`
A	All conformance tests in a single conformance test class in a conformant standard *SHALL* have the same standardization target.

This means that all requirements are considered as targeting the same entity being tested for a particular certificate of conformance. The test may specify other types as intermediaries or indirect dependencies (see indirect dependency of-a-requirements-class).

Table 5

PER005	**/per/core/repeated-requirements** If needed, a requirement *MAY* be repeated word for word in another requirement up to but not including the identification of the standardization target type.

This second statement will be in a separate requirements class, since it will have a separate standardization target and thus belong to the requirements to be tested by a separate conformance class. For example, in a service interface, a standard may be written that requires both the client and server to use a particular language for data transmission. Since the client and server are different standardization targets types (except in some special circumstances), they will have different conformance test classes.

One solution is to state the requirement twice, once for each target. The most common alternative is to introduce a new "superclass".

Table 6

PER006	**/per/core/abstract-superclass** The standard *MAY* introduce an abstract superclass of all affected standardization target types and use this for the requirements common to all of the affected target types. This is diagrammed in Figure 1.

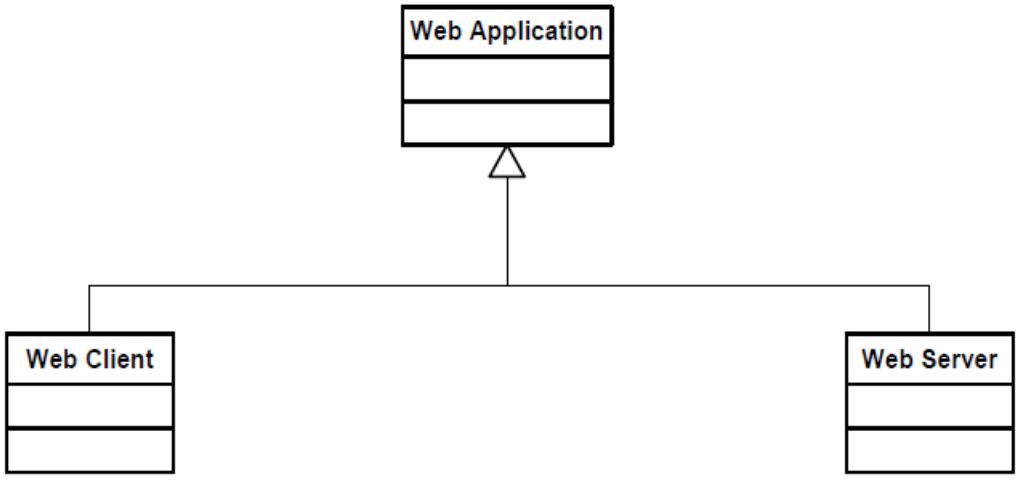

Figure 1 — Abstract superclass example

Example — Abstract Superclass:

8.1.2. The “standards” document

Each standard document is comprised of a set of requirements and their associated conformance tests. Requirements are grouped into requirements classes. Each requirements class is contained within one section/clause in a standards document.

REQUIREMENT 6	
IDENTIFIER	`/req/core/requirements-grouped`
A	Requirements *SHALL* be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements classes.

The following requirement states that the sequence of requirements and requirements classes is the same as the sequence of conformance tests and conformance classes in the conformance suite.

REQUIREMENT 7	
IDENTIFIER	`/req/core/requirements-test-suite-structure`
A	The requirements structure of the document *SHALL* be in a logical correspondence with the test suite structure.

If two requirements are in the same requirments class, they should be tested in the same conformance class in the conformance suite. Each requirement is separately identifiable either by a label as is done in the ModSpec.

In summary, the structure of the requirements and requirements classes of the model should be reflected in the organization of the conformance tests and classes, and also in the structure of the normative clauses in the specification document.

8.1.3. Conformance Test Suite

The requirements specified in this clause will be applied directly to the test suite, and in particular to the conformance classes. By definition, a “test suite” is a collection of identifiable conformance classes. A conformance class is a well-defined set of conformance tests. Each conformance test is a concrete or abstract (depending on the type of suite) description of a test to be performed on each candidate conformant implementation, to determine if it meets a well-defined set of requirements as stated in the normative clauses of the standards document.

NOTE: The Test Suite is normative in the sense that it describes the tests to be performed to pass conformance, but it specifies no requirements in any other sense. The requirements are specified in the body of the standard. The test suite only describes in detail how those requirements should be tested.

In each of the profiles defined in the Clauses to follow, some set of entities, types, elements, or objects are defined and segregated into implementation requirements classes.

REQUIREMENT 8	
IDENTIFIER	`/req/core/requirements-class-correspondence-to-conformance-classes`
A	The requirements classes *SHALL* be in a one-to-one correspondence to the conformance test classes, and thus to the various certificate of conformance types possible for a candidate implementation.

Strict parallelism of implementation and governance is the essence of this standard.

8.1.4. Requirements for Modularity

8.1.4.1. Each Conformance class tests a complete requirements class

REQUIREMENT 9	
IDENTIFIER	`/req/core/no-optional-tests`
A	A conformance class *SHALL* not contain any optional conformance tests.

This requirement stops conformance classes from containing optional requirements and tests, and, at least as far as the standard is concerned, makes all certificates of conformance mean that exactly the same tests have been conducted. Standards documents may use recommendations for such options, but the conformance test classes do not test recommendations.

Table 7

PER007	**/per/core/conf-class-paramterized** A Conformance class *MAY* be parameterized.

This means that the class's tests depend on some parameter that must be defined before the tests can be executed. This can be thought of as an "if-then-else" decision tree.

For example, if a conformance class needs to apply tests against a specific data format, such as GML or KML, then XYZ(GML) is XYZ using GML, and XYZ(KML) is XYZ using KML. Because the parameters choose which requirements will be tested, two conformance classes with distinct parameters should be considered as distinct conformance classes.

The most common parameters are the identities of indirect dependencies. For example, if a service uses or produces feature data, the format of that data may be a parameter, such as GML, KML or GeoJSON. When reading a certificate of conformance, the values of such parameters are very important.

REQUIREMENT 10	
IDENTIFIER	`/req/core/all-parameters-expressed`
A	A certificate of conformance *SHALL* specify all parameter values used to pass the tests in its conformance test class.

Conformance to a particular conformance class means exactly the same thing everywhere.

REQUIREMENT 11	
IDENTIFIER	`/req/core/conf-class-single-req-class`
A	A Conformance class *SHALL* explicitly test only requirements from a single requirements class.

This means that there is a strict correspondence between the requirements classes and the conformance test classes in the test suite. Recall that a conformance test class may specify dependencies causing other conformance test classes to be used, but this is a result of an explicit requirement in the "home" requirements class.

REQUIREMENT 12

IDENTIFIER	`/req/core/con-class-dependencies`
A	A Conformance class *SHALL* specify any other conformance class upon which it is dependent and that other conformance class shall be used to test the specified dependency.

Such referenced conformance classes may be in the same standard or may be a conformance class of another standard.

Example — Indirect dependency on schema: If a service specifies that a particular output is required to be conformant to a conformance test class in a specific standard (say a normatively referenced XML schema), then the conformance class of that normative reference will be used to test that output. For example, if an OGC Web Feature Service (WFS) implementation instance specifies that its feature collection output is compliant to a particular profile of GML, then that profile of GML will be used to validate that output. This means that the service is indirectly tested using the GML standard. In other words, GML is an indirect dependency of the original service.

Requirements classes may be optional as a whole, but not piecemeal. This means that every implementation that passed a particular conformance class satisfies exactly the same requirements and passes exactly the same conformance tests. Differences between implementations will be determined by which conformance test classes are passed, not by listing of which options within a class were tested. If a standard's authors wish to make a particular requirement optional, Requirement 9: `/req/core/no-optional-tests` forces them to include it in a separate requirements class (and therefore in a separate conformance test class) which can be left untested.

NOTE: Standards developed outside the OGC may not follow a strict parallelism between requirement specification and testing, so for use within a standard compliant to the ModSpec, special care must be taken in importing conformance test classes from other standards.

REQUIREMENT 13

IDENTIFIER	`/req/core/imported-requirements-class`
A	If a requirements class is imported from another standard for use within a standard conformant to the ModSpec, and if any imported requirement is "optional," then that requirement SHALL be factored out as a separate requirements class in the profile of that imported standard used in the conformant standard.
B	Each such used requirements class SHALL be a conformance class of the source standard or a combination of conformance classes of the source standard or standards.

The tracking of the parallelism between requirements and tests should be easy if the standards document is non-ambiguous. To insure this, by utilizing the names of the two types of classes the following requirement places a default mapping between the two.

REQUIREMENT 14	
IDENTIFIER	`/req/core/all-classes-explicitly-named`
A	For the sake of consistency and readability, all requirements classes and all conformance test classes *SHALL* be explicitly named, with corresponding requirements classes and conformance test classes having similar names.

Logically, a requirements class (set of requirements) and a conformance class (set of tests) are not comparable. This can be remedied by noting that both have a consistent relation to a set of requirements. A requirements class is a set of requirements. A conformance class tests a set of requirements. Therefore a requirements class corresponds precisely to a conformance class if they both are related (as described) to the same set of requirements.

8.1.5. Requirements classes contain all requirements tested by a conformance test case

REQUIREMENT 15	
IDENTIFIER	`/req/core/req-in-only-one-req-class`
A	Each requirement in the standard *SHALL* be contained in one and only one requirements class.
B	Inclusion of any requirement in a requirements class by a conformance class *SHALL* imply inclusion of all requirements in its class (as a dependency).

Unless a requirement is referenced in a conformance test and thus in a conformance class, it cannot be considered a requirement since no test has been defined for it.

RECOMMENDATION 2	
IDENTIFIER	`/rec/core/parallel-structure`
A	If possible, the structure of the normative clauses of the standard *SHOULD* parallel the structure of the conformance classes in the conformance clause.

The above requirement in conjunction with Requirement 9: `/req/core/no-optional-tests` means that all requirements in a conformant standard will be tested in some conformance class. In the best example, a requirement should be contained explicitly in one and only one requirements class and tested in one and only one conformance class. This is not really a requirement here, since a single requirement can be stated twice in different requirements classes.

REQUIREMENT 16	
IDENTIFIER	`/req/core/co-dependent-requirements`
A	If any two requirements are co-dependent (each dependent on the other) then they *SHALL* be in the same requirements class.
B	If any two requirements classes are co-dependent, they *SHALL* be merged into a single class.

Normally, circular dependencies between implementation components are signs of a poor design, but they often cannot be avoided because of other considerations (code ownership for example).

RECOMMENDATION 3	
IDENTIFIER	`/rec/core/circular-dependencies`
A	Circular dependencies of all types *SHOULD* be avoided whenever possible.

REQUIREMENT 17	
IDENTIFIER	`/req/core/structure-requirements-classes`
A	There *SHALL* be a natural structure to the requirements classes so that each may be implemented on top of any implementations of its dependencies and independent of its extensions.

NOTE: The only certain manner to test this requirement maybe to create a reference implementation.

This requirement is more important and may be more difficult than it seems. It states simply that conformance classes and their associated requirements classes can be put in a one-to-one correspondence to a fully modular implementation of the complete standard (at least against a single standardization target). Implementors who wish to sacrifice modularity for some other benefit can still do what they want; the requirement here only states that if the software requirements classes are properly separated, they can be implemented in a “plug’n’play” fashion.

REQUIREMENT 18	
IDENTIFIER	`/req/core/requirements-and-dependencies`
A	No requirements class *SHALL* redefine the requirements of its dependencies, unless that redefinition is for an entity derived from but not contained in those dependencies.

This means, for example, that a UML classifier cannot be redefined in a new extension. If a new version of the classifier is needed it has to be a valid subtype of the original.

In terms of generalization, subclassing, extension and restriction (into a new class or type) are all acceptable, redefinition (of an old class or type) is not.

Clause 8.1.2 makes some pointed suggestion as to how to organize the conformance classes and normative clauses in parallel to make this requirement easier to verify.

Most standards include examples, which are useful for illustrative or pedagogical purposes. However, it is not possible to write a standard "by example" that leads to conformance tests. Examples are therefore non-normative, by definition.

8.1.6. Profiles are defined as sets of conformance classes

All the conformance classes created in a standard form a base (an upper bound of all conformance classes) for defining profiles as defined in ISO/IEC 10000 (see ISO/IEC DIR 2). The base for creating a profile can be defined as the union of all requirements classes.

REQUIREMENT 19	
IDENTIFIER	`/req/core/profile-conformance`
A	The conformance tests for a profile of a standard *SHALL* be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets.

8.1.7. There is a Defined Core

The following requirements define the content of the core.

REQUIREMENT 20	
IDENTIFIER	`/req/core/core-requirements-separate`
A	Every standard *SHALL* define and identify a core set of requirements as a separate requirements class with a corresponding conformance class.

The following states that any recommendations applicable to the entire standard are in the core document.

REQUIREMENT 21	
IDENTIFIER	`/req/core/general-recommendations-core`

REQUIREMENT 21	
A	All general recommendations for a standard *SHALL* be in the core.

The following states that all non-core requirements classes have a standardization target type that is a sub-type of the core.

REQUIREMENT 22	
IDENTIFIER	`/req/core/req-class-not-core-stt-subtype-of-core`
A	Every requirements class in a standard, with the exception of the core, *SHALL* have a standardization target type which is a subtype of that of the core.
B	Every requirement class, with the exception of the core, *SHALL* have the core as a direct dependency.

The following recommendation is guidance to the group developing a standard to keep the core as simple as possible.

RECOMMENDATION 4	
IDENTIFIER	`/rec/core/simple-core`
A	The core *SHOULD* be as simple as possible.

The following enables the development and documentation of anstract standards. Examples of such standards are the OGC Abstract Specification. Typically, an abstract standard cannot be directly implemented.

Table 8

PER008	**/per/core/core-type** The core *MAY* be partially or totally abstract.

Table 9

PER009	**/per/core/req-class-another-standard** The core requirements class *MAY* be a conformance class in another standard.

RECOMMENDATION 5	
IDENTIFIER	`/rec/core/optional-tests`
A	If a requirements class is from another standard, the current standard *SHOULD* identify any optional tests in that conformance class that are required by the current standard's core requirements class. See Requirement 13: `/req/core/imported-requirements-class`.

Since the core requirements class is contained (as a direct dependency) in each other requirements class with a similar standardization target type, the general recommendations are thus universal to all requirements classes.

Table 10

PER010	**/per/core/core-maybe-recommendations** Since the basic concept of some standards is mechanism not implementation, the core *MAY* contain only recommendations, and include no requirements.

NOTE: In most cases, if someone feels the need to define a "simple" version of the standard, it is probably a good approximation of the best core. For example, the core of a refactored GML might be the equivalent of the "GML for Simple Features" profile. The core for any SQL version of feature geometry is probably "Simple Features."

8.1.8. Extensions are requirements classes

A common mechanism to extend the functionality of a standard is to define extensions, which may be either local or encompass other standards.

Standards should use extensions as required and feasible, but should never hinder them.

REQUIREMENT 23	
IDENTIFIER	`/req/core/core-and-extensions`
A	Each standard conformant to the ModSpec *SHALL* consist of a core and some number of requirements classes defined as extensions to that core.

REQUIREMENT 24	
IDENTIFIER	`/req/core/extensions-conformant-to-the-modspec`
A	A standard conformant to the ModSpec *SHALL* require all conformant extensions to be conformant to the ModSpec.

Since software is evolutionary at its best, it would not be wise to restrict that evolutionary tendency by restricting the specification of extensions. A good standard will thus list the things a standardization target has to do, but will never list things that a standardization target might want to do above and beyond the current design requirements.

REQUIREMENT 25	
IDENTIFIER	`/req/core/restriction-of-extensions`
A	A standard conformant to the ModSpec *SHALL* never restrict in any manner future, logically valid extensions of its standardization target types.

The above requirement should not be interpreted as a restriction on quality control. Any efforts by a standard to enforce a level of quality on its standardization targets, when well and properly formed, do not interfere with the proper extension of those targets. So, the standard may require its standardization targets to behave in a certain manner when presented with a logical inconsistency, but that inconsistency must be fundamental to the internal logic of the model, and not a possible extension. Thus, a standard may require a standardization target to accept GML as a feature specification language, but cannot require a standardization target to not accept an alternative, such as KML, or GeoJSON, as long at that alternative can carry viable information consistent with the fundamental intent of the standard.

8.1.9. Optional requirements are organized as optional requirements classes

REQUIREMENT 26	
IDENTIFIER	`/req/core/optional-requirements`
A	The only conditional (optional) requirements acceptable in a standard conformant with the ModSpec *SHALL* be expressible as a list of conformance classes to be passed.

A standard and implementations of that standard are restricted by this requirement.

Requirements of the form “if the implementation does this, it must do it this way” are considered to be options and should be in a separate requirements class.

8.1.10. Requirements classes intersect overlap only by reference

REQUIREMENT 27	
IDENTIFIER	`/req/core/req-class-overlap-by-reference`

REQUIREMENT 27	
A	The common portion of any two requirements classes *SHALL* consist only of references to other requirements classes.

This implies that each requirement is directly in exactly one requirements class and all references to that requirement from another requirements class must include its complete "home" requirements class. This means that requirements for dependencies will often result in conformance test cases which require the execution of the dependency conformance class. See for example [annex-A-2-1].

NOTE: All general recommendations are in the core requirements class. The core conformance test class contains tests that all other conformance classes must pass.

9

MAPPING THE MODSPEC TO TYPES OF MODELS

9 MAPPING THE MODSPEC TO TYPES OF MODELS

9.1. Semantics

The previous section defines requirements for conformance to the ModSpec However, testing for that conformance may depend on how the various forms and parts of a conformant standard are viewed. Additional Parts to the ModSpec Standard specify how those views are to be defined. Standards that take an alternative mechanism to the ones defined in the additional Parts must be tested solely on the structure of their conformance test suite until such times as an extension to the ModSpec is defined for that alternate mechanism.

Standards are often structured about some form of modeling language, or implementation paradigm. Additional Parts to the ModSpec define a mechanism to map parts of the model (language, schema, etc.) to the conformance classes used as the model from Clause 6.1.

As suggested in Clause Requirement 22: `/req/core/req-class-not-core-stt-subtype-of-core`, the structure of the normative clauses in a standard should parallel the structure of the conformance test classes in that standard. The structure of the normative clauses in a well written standard will follow the structure of its model. This means that all three are parallel.

9.2. A Note on Data Models

If a data model is to be used to define the parameters of operational interfaces, then that model should belong in the core since it can be considered as part of a common reference model and vocabulary.

If a data model is to be used to create “data transfer” elements, the issue is more complex. In the use of parameter names and types in the operational model above, the definition of a common vocabulary in the core is justifiable. In the case where data transfer elements are being defined, it may be that some types and elements are a defining separator between conformance classes and have to exist independently of such data elements defined for non-dependent classes. For these reasons, care should be taken in creating separable data transfer schemas across requirements. Dependencies in the schemas will have to parallel the dependencies in the requirements classes. The mechanism for enforcing this is dependent on the schema language.

A

ANNEX A (NORMATIVE) ABSTRACT CONFORMANCE TEST SUITE

A

ANNEX A (NORMATIVE) ABSTRACT CONFORMANCE TEST SUITE

A.1. Conformance Test Class: The Core

CONFORMANCE CLASS A.1	
IDENTIFIER	`https://www.opengis.net/spec/modspec-1/2.0/conf/core`
REQUIREMENTS CLASS	https://www.opengis.net/spec/modspec-1/2.0/req/req-class-core
TARGET TYPE	Modular Standard
CONFORMANCE TESTS	Abstract test A.1: `/conf/core/reqs-are-in-class` Abstract test A.2: `/conf/core/reqs-are-testable` Abstract test A.3: `/conf/core/all-components-assigned-uri` Abstract test A.4: `/conf/core/vocabulary-and-parent-req-class` Abstract test A.5: `/conf/core/single-standardization-target-type` Abstract test A.6: `/conf/core/test-class-single-standardization-target-type` Abstract test A.7: `/conf/core/requirements-grouped` Abstract test A.8: `/conf/core/requirements-test-suite-structure` Abstract test A.9: `/conf/core/requirements-class-correspondence-to-conformance-classes` Abstract test A.10: `/conf/core/no-optional-tests` Abstract test A.11: `/conf/core/all-parameters-expressed` Abstract test A.12: `/conf/core/conf-class-single-req-class` Abstract test A.13: `/conf/core/con-class-dependencies` Abstract test A.14: `/conf/core/imported-requirements-class` Abstract test A.15: `/conf/core/all-classes-explicitly-named` Abstract test A.16: `/conf/core/req-in-only-one-req-class` Abstract test A.17: `/conf/core/co-dependent-requirements` Abstract test A.18: `/conf/core/structure-requirements-classes` Abstract test A.19: `/conf/core/requirements-and-dependencies` Abstract test A.20: `/conf/core/profile-conformance`

CONFORMANCE CLASS A.1	
	Abstract test A.21: `/conf/core/core-requirements-separate` Abstract test A.22: `/conf/core/requirements-and-dependencies-2` Abstract test A.23: `/conf/core/requirements-and-dependencies-3` Abstract test A.24: `/conf/core/core-and-extensions` Abstract test A.25: `/conf/core/extensions-conformant-to-the-modspec` Abstract test A.26: `/conf/core/restriction-of-extensions` Abstract test A.27: `/conf/core/optional-requirements` Abstract test A.28: `/conf/core/req-class-overlap-by-reference`

A.1.1. Requirements are in class

ABSTRACT TEST A.1	
IDENTIFIER	`/conf/core/reqs-are-in-class`
REQUIREMENT	/req/core/reqs-are-in-class
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that each requirement in a standard *SHALL* be associated with exactly one requirements class.
TEST METHOD	Inspect the document to verify the above.

A.1.2. Requirements are testable

All the parts of a requirement *SHALL* be testable.

ABSTRACT TEST A.2	
IDENTIFIER	`/conf/core/reqs-are-testable`
REQUIREMENT	Requirement 1: `/req/core/reqs-are-testable`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance

ABSTRACT TEST A.2	
TEST PURPOSE	Validate that all the parts of a requirement *SHALL* be testable and that Failure to pass any part of a requirement *SHALL* be a failure to pass the associated conformance test.
TEST METHOD	Inspect the document to verify the above.

A.1.3. All components have an assigned URI

ABSTRACT TEST A.3	
IDENTIFIER	`/conf/core/all-components-assigned-uri`
REQUIREMENT	Requirement 2: `/req/core/all-components-assigned-uri`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that each component of a standard, including requirements, requirements modules, requirements classes, conformance test, conformance modules, and conformance test classes *SHALL* be assigned a unique identifer or label.
TEST METHOD	Inspect the document to verify the above.

A.1.4. Requirements on vocabulary are appropriately placed

ABSTRACT TEST A.4	
IDENTIFIER	`/conf/core/vocabulary-and-parent-req-class`
REQUIREMENT	Requirement 3: `/req/core/vocabulary-and-parent-req-class`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that requirements on the use and interpretation of vocabulary *SHALL* be in the requirements class where that use or interpretation is used.
TEST METHOD	Inspect the document to verify the above.

A.1.5. Requirements have a single target type

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/core/single-standardization-target-type`
REQUIREMENT	Requirement 4: `/req/core/single-standardization-target-type`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that each requirement class in a conformant standard *SHALL* have a single standardization target type.
TEST METHOD	Inspect the document to verify the above.

A.1.6. Conformance test classes have a single target type

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/core/test-class-single-standardization-target-type`
REQUIREMENT	/req/core/test-class-single-standardization-target-type
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that all conformance tests in a single conformance test class in a conformant standard *SHALL* have the same standardization target type.
TEST METHOD	Inspect the document to verify the above.

A.1.7. Requirements are organized by clauses

ABSTRACT TEST A.7	
IDENTIFIER	`/conf/core/requirements-grouped`
REQUIREMENT	Requirement 6: `/req/core/requirements-grouped`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that requirements *SHALL* be grouped together in clauses (numbered sections) of the document in a strictly hierarchical manner, consistent with requirements classes.
TEST METHOD	Inspect the document to verify the above.

A.1.8. Conformance test classes are consistent with requirements classes

ABSTRACT TEST A.8	
IDENTIFIER	`/conf/core/requirements-test-suite-structure`
REQUIREMENT	Requirement 7: `/req/core/requirements-test-suite-structure`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that the requirements structure of the document *SHALL* be in a logical correspondence with the test suite structure.
TEST METHOD	Inspect the document to verify the above.

A.1.9. Requirement classes and the Conformance Test classes in correspondence

ABSTRACT TEST A.9	
IDENTIFIER	`/conf/core/requirements-class-correspondence-to-conformance-classes`
REQUIREMENT	Requirement 8: `/req/core/requirements-class-correspondence-to-conformance-classes`

ABSTRACT TEST A.9	
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that the requirements classes *SHALL* be in a one-to-one correspondence to the conformance test classes, and thus to the various certificate of conformance types possible for a candidate implementation.
TEST METHOD	Inspect the document to verify the above.

A.1.10. No Optional Elements in Requirements classes

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/core/no-optional-tests`
REQUIREMENT	Requirement 9: `/req/core/no-optional-tests`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that a conformance class *SHALL* not contain any optional conformance tests.
TEST METHOD	Inspect the document to verify the above.

A.1.11. Certificate of conformance specifies all parameters used

ABSTRACT TEST A.11	
IDENTIFIER	`/conf/core/all-parameters-expressed`
REQUIREMENT	Requirement 10: `/req/core/all-parameters-expressed`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance

ABSTRACT TEST A.11	
TEST PURPOSE	Validate that a certificate of conformance *SHALL* specify all parameter values used to pass the tests in its conformance test class.
TEST METHOD	Inspect the document to verify the above.

A.1.12. Conformance class tests only one requirements class

ABSTRACT TEST A.12	
IDENTIFIER	`/conf/core/conf-class-single-req-class`
REQUIREMENT	Requirement 11: `/req/core/conf-class-single-req-class`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that a Conformance class *SHALL* explicitly test only requirements from a single requirements class.
TEST METHOD	Inspect the document to verify the above.

A.1.13. Conformance class specifies all dependencies

ABSTRACT TEST A.13	
IDENTIFIER	`/conf/core/con-class-dependencies`
REQUIREMENT	Requirement 12: `/req/core/con-class-dependencies`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that a Conformance class *SHALL* specify any other conformance class upon which it is dependent and that other conformance class shall be used to test the specified dependency.
TEST METHOD	Inspect the document to verify the above.

A.1.14. Imported Conformance class tests are consistent with the specification

ABSTRACT TEST A.14	
IDENTIFIER	`/conf/core/imported-requirements-class`
REQUIREMENT	Requirement 13: `/req/core/imported-requirements-class`
TEST PURPOSE	Validate that if a requirements class is imported from another standard for use within a standard conformant to the ModSpec, and if any imported requirement is “optional,” then that requirement SHALL be factored out as a separate requirements class in the profile of that imported standard used in the conformant standard.
DESCRIPTION	And that each such used requirements class SHALL be a conformance class of the source standard or a combination of conformance classes of the source standard or standards.. Test Method:: Inspect the document to verify the above. Test Type:: Conformance

A.1.15. Naming consistency

ABSTRACT TEST A.15	
IDENTIFIER	`/conf/core/all-classes-explicitly-named`
REQUIREMENT	Requirement 14: `/req/core/all-classes-explicitly-named`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that all requirements classes and all conformance test classes *SHALL* be explicitly named, with corresponding requirements classes and conformance test classes having similar names.
TEST METHOD	Inspect the document to verify the above.

A.1.16. Requirements in one and only one requirements class

ABSTRACT TEST A.16	
IDENTIFIER	`/conf/core/req-in-only-one-req-class`
REQUIREMENT	Requirement 15: `/req/core/req-in-only-one-req-class`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that each requirement in the standard *SHALL* be contained in one and only one requirements class. And that Inclusion of any requirement in a requirements class by a conformance class *SHALL* imply inclusion of all requirements in its class (as a dependency).
TEST METHOD	Inspect the document to verify the above.

A.1.17. Co-dependent Requirements are in the same requirements class

ABSTRACT TEST A.17	
IDENTIFIER	`/conf/core/co-dependent-requirements`
REQUIREMENT	Requirement 16: `/req/core/co-dependent-requirements`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that if any two requirements are co-dependent (each dependent on the other) then they *SHALL* be in the same requirements class. And if any two requirements classes are co-dependent, they *SHALL* be merged into a single class.
TEST METHOD	Inspect the document to verify the above.

A.1.18. Modularity in implementation is possible

ABSTRACT TEST A.18	
IDENTIFIER	`/conf/core/structure-requirements-classes`
REQUIREMENT	Requirement 17: `/req/core/structure-requirements-classes`

ABSTRACT TEST A.18	
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that there *SHALL* be a natural structure to the requirements classes so that each may be implemented on top of any implementations of its dependencies and independent of its extensions.
TEST METHOD	Inspect the document to verify the above.

A.1.19. Requirements follow rules of inheritance

ABSTRACT TEST A.19	
IDENTIFIER	`/conf/core/requirements-and-dependencies`
REQUIREMENT	Requirement 18: `/req/core/requirements-and-dependencies`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that no requirements class *SHALL* redefine the requirements of its dependencies, unless that redefinition is for an entity derived from but not contained in those dependencies.
TEST METHOD	Inspect the document to verify the above.

A.1.20. Profiles are expressed as sets of conformance classes

ABSTRACT TEST A.20	
IDENTIFIER	`/conf/core/profile-conformance`
REQUIREMENT	Requirement 19: `/req/core/profile-conformance`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance

ABSTRACT TEST A.20	
TEST PURPOSE	Validate that the conformance tests for a profile of a standard *SHALL* be defined as the union of a list of conformance classes that are to be satisfied by that profile's standardization targets.
TEST METHOD	Inspect the document to verify the above.

A.1.21. There is a named Core requirements class

ABSTRACT TEST A.21	
IDENTIFIER	`/conf/core/core-requirements-separate`
REQUIREMENT	Requirement 20: `/req/core/core-requirements-separate`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that every standard *SHALL* define and identify a core set of requirements as a separate requirements class with a corresponding conformance class.
TEST METHOD	Inspect the document to verify the above.

A.1.22. General conditions are in the core

ABSTRACT TEST A.22	
IDENTIFIER	`/conf/core/requirements-and-dependencies-2`
REQUIREMENT	/req/core/requirements-and-dependencies-2
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that all general recommendations for a standard *SHALL* be in the core.
TEST METHOD	Inspect the document to verify the above.

A.1.23. Every extension has a consistent target type

ABSTRACT TEST A.23	
IDENTIFIER	`/conf/core/requirements-and-dependencies-3`
REQUIREMENT	/req/core/requirements-and-dependencies-3
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that every requirements class in a standard, with the exception of the core, *SHALL* have a standardization target type which is a subtype of that of the core. And that every requirement class, with the exception of the core, *SHALL* have the core as a direct dependency.
TEST METHOD	Inspect the document to verify the above.

A.1.24. A standard is a core plus some number of extensions

ABSTRACT TEST A.24	
IDENTIFIER	`/conf/core/core-and-extensions`
REQUIREMENT	Requirement 23: `/req/core/core-and-extensions`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that each standard conformant to the ModSpec *SHALL* consist of a core and some number of requirements classes defined as extensions to that core.
TEST METHOD	Inspect the document to verify the above.

A.1.25. Conformance to this ModSpec is required for any extensions

ABSTRACT TEST A.25	
IDENTIFIER	`/conf/core/extensions-conformant-to-the-modspec`
REQUIREMENT	Requirement 24: `/req/core/extensions-conformant-to-the-modspec`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that a standard conformant to the ModSpec *SHALL* require all conformant extensions to be conformant to the ModSpec.
TEST METHOD	Inspect the document to verify the above.

A.1.26. Future extensions cannot be restricted

ABSTRACT TEST A.26	
IDENTIFIER	`/conf/core/restriction-of-extensions`
REQUIREMENT	Requirement 25: `/req/core/restriction-of-extensions`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that a standard conformant to the ModSpec *SHALL* never restrict in any manner future, logically valid extensions of its standardization target types.
TEST METHOD	Inspect the document to verify the above.

A.1.27. Optional requirements are organized as requirements classes

ABSTRACT TEST A.27	
IDENTIFIER	`/conf/core/optional-requirements`
REQUIREMENT	Requirement 26: `/req/core/optional-requirements`

ABSTRACT TEST A.27	
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that the only conditional (optional) requirements acceptable in a standard conformant with the ModSpec *SHALL* be expressible as a list of conformance classes to be passed.
TEST METHOD	Inspect the document to verify the above.

A.1.28. Requirements classes intersect overlap only by reference

ABSTRACT TEST A.28	
IDENTIFIER	`/conf/core/req-class-overlap-by-reference`
REQUIREMENT	Requirement 27: `/req/core/req-class-overlap-by-reference`
TEST METHOD	Inspect the document to verify the above.
TEST TYPE	Conformance
TEST PURPOSE	Validate that the common portion of any two requirements classes *SHALL* consist only of references to other requirements classes.
TEST METHOD	Inspect the document to verify the above.

B

ANNEX B (NORMATIVE) OGC ONLY: CHANGES REQUIRED IN THE OGC STANDARDS

B

ANNEX B (NORMATIVE) OGC ONLY: CHANGES REQUIRED IN THE OGC STANDARDS

NOTE: The following is for OGC Standards and Abstract Specifications only: No changes are required to existing OGC Standards

B.1. New OGC standards and revisions to existing OGC standards

Any new standard or major revision of an existing standard *SHALL* comply with the ModSpec in the structure of its internal models and its conformance tests.

Failure to conform by a candidate standard to the ModSpec should be specifically noted and reasons given for such non-compliance in the conformance clauses of any new or new version of such candidate standards.

The adoption of such documents not compliant with the ModSpec *SHALL* be considered as an authorized exception to the requirements of the ModSpec by the approporiate authority, such as the OGC or ISO. An exception to the rules of the authority such as the OGC and will require a two-thirds (2/3) majority ("Robert's Rules") or as specified in the authorities Policy and Procedures for an exception to procedure. In the OGC, a similar vote is required within the Executive Planning Committee or as specified in any Policy and Procedure document used by this committee.

C

ANNEX C (INFORMATIVE) MODSPEC UML MODEL, SEMANTICS, AND DEFINITIONS

C

ANNEX C (INFORMATIVE) MODSPEC UML MODEL, SEMANTICS, AND DEFINITIONS

C.1. Semantically ordered definitions

Clause 4 formally defines the terms used in the conformance tests in alphabetical order. It may be easier to understand the more significant terms in the following less formal definitions arranged in a bottom-up order:

1. a *standardization target type* is a type of entity about which a standard is written. An instance of a *standardization target type* is a *standardization target*. A standard may address multiple targets in separate *conformance classes*.
2. a *requirement* is a statement of a condition to be satisfied by a single *standardization target type*, and it must be stated in "normative" language.
3. a *conformance test* checks if a set of *requirements* are met (**pass**) or not met (**fail**) by a *standardization target*.
4. all *conformance tests* are graded as **pass** or **fail** against each instance of the *standardization target type*.
5. a *requirement* is associated to one *conformance test*.
6. a *recommendation* is a suggestion and is not associated to any *conformance test*.
7. a *requirements class* is a set of one or more *requirements* all with the same *standardization target type*.
8. a *conformance (test) class* is a collection of *conformance tests* that are associated to and only to the requirements in a corresponding *requirements class*.
9. a *conformance (test) module* is a reusable collection of related *conformance test classes*.
10. a **conformant implementation** is a *standardization target* that has successfully passed all tests in a specified *conformance (test) class* and received a *certificate of conformance*.

11. the *Core requirements class* of a standard is the minimal set of *requirements* which must be supported by all **conformant implementations**. If a standard addresses multiple *standardization target types*, it may have a **core** for each **target type**.

12. an **extension** of a *requirements class* is an additional *requirements class* (the extension) that adds additional *requirements* to the first *requirements class* (the **base requirements class** being extended). The extension is said to be dependent on the **base**. Any *conformance (test) class* must identify all its dependencies during the execution of conformance tests against a candidate *standardization target*.

C.2. UML Model

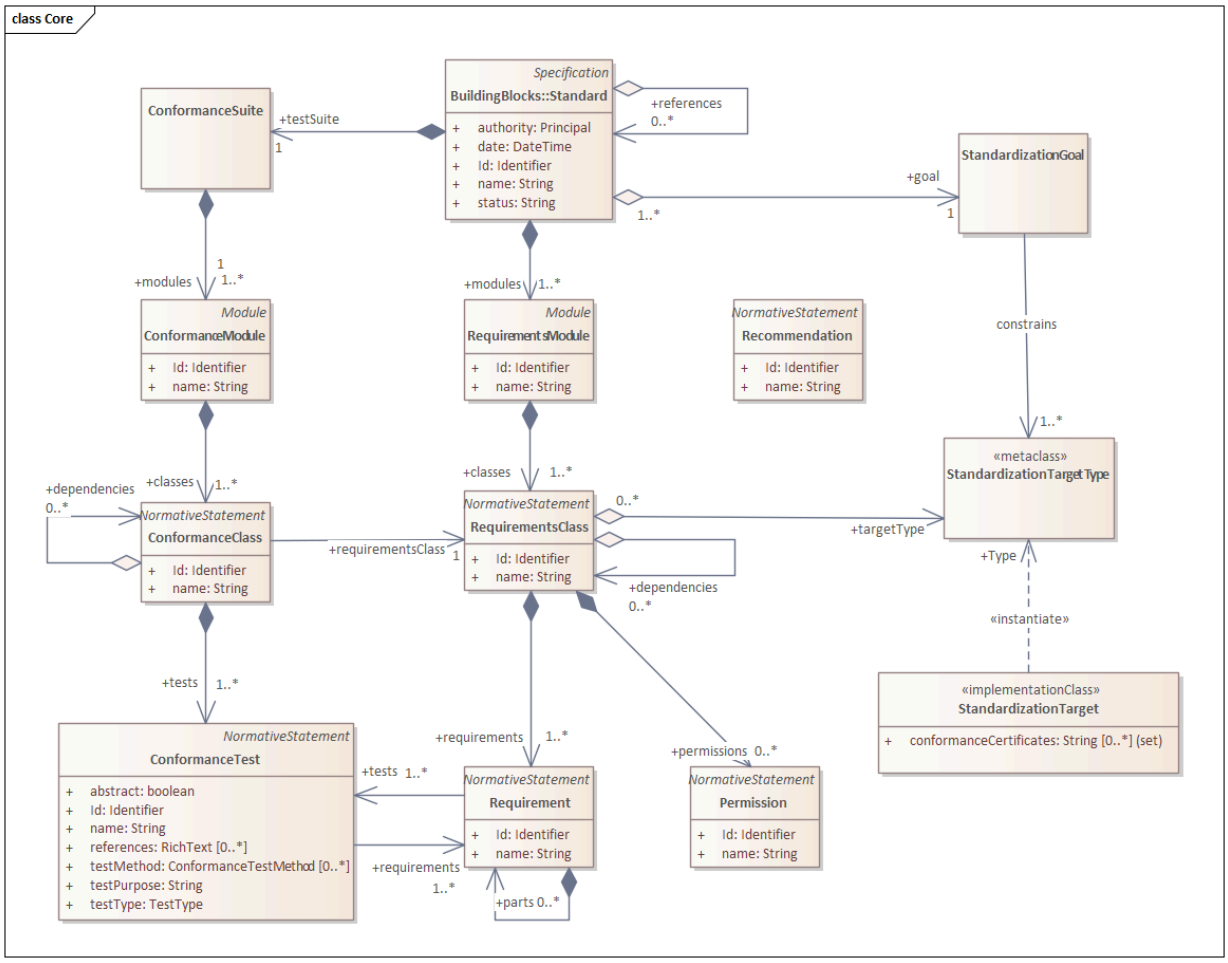

Figure C.1

Figure 1 represents a UML model consistent with the model described in Clause 8 of this document. The following subclauses describe the classes shown in this UML class diagram.

D

ANNEX D (NORMATIVE) ACKNOWLEDGEMENTS

D

ANNEX D (NORMATIVE) ACKNOWLEDGEMENTS

The following OGC Members were key contributors to Version 1 of the ModSpec

PERSON	ORGANIZATION REPRESENTED
Simon Cox	CSIRO
David Danko	Esri
James Greenwood	SeiCorp, Inc.
John R. Herring	Oracle USA
Andreas Matheus	University of the Bundeswehr — ITS
Richard Pearsall	US National Geospatial-Intelligence Agency (NGA)
Clemens Portele	interactive instruments GmbH
Barry Reff	US Department of Homeland Security (DHS)
Paul Scarponcini	Bentley Systems, Inc.
Arliss Whiteside	BAE Systems — C3I Systems

BIBLIOGRAPHY

BIBLIOGRAPHY

To preserve a unique numeric identifier for all documents listed as references in this standard, the numbering of references in this annex is continued from the list of normative reference in Clause 3.

[1] ISO/IEC: ISO/IEC 9075:2003, *ISO/IEC JTC 1, ISO/IEC 9075:2003 — Information Technology — Database Languages — SQL.*. ISO, IEC (2003).

[2] ISO/IEC: ISO/IEC TR 10000, *ISO/IEC TR 10000: Information Technology — Framework and taxonomy of International Standardized Profiles*. ISO, IEC

[3] ISO/IEC: ISO/IEC 13249-3:2006, *Information technology — Database languages — SQL multimedia and application packages — Part 3: Spatial*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2006). https://www.iso.org/standard/38651.html.

[4] W3C: W3C xmlschema11-1, *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. World Wide Web Consortium http://www.w3.org/TR/xmlschema11-1/.

[5] W3C: W3C xmlschema11-2, *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. World Wide Web Consortium http://www.w3.org/TR/xmlschema11-2/.