OGC® DOCUMENT: 24-065

External identifier of this OGC® document: http://www.opengis.net/doc/spec/modspec-2/1.1

Open Geospatial Consortium

THE MODSPEC MODEL - PART 2: UML

STANDARD
Implementation

DRAFT

Version: 1.1.0
Submission Date: 2025-04-15
Approval Date: 2029-03-30
Publication Date: 2029-03-30
Editor: Carl Reed, John Herring

License Agreement

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

I. ABSTRACT v

II. KEYWORDS v

III. PREFACE vi

IV. SECURITY CONSIDERATIONS vii

V. SUBMITTING ORGANIZATIONS viii

VI. SUBMITTERS viii

VII. REVISION HISTORY viii

VIII. FUTURE WORK viii

IX. CONTRIBUTORS ix

X. ACKNOWLEDGEMENTS ix

1. SCOPE 2
   1.1. Introduction 2

2. CONFORMANCE 4

3. NORMATIVE REFERENCES 6

4. TERMS AND DEFINITIONS 8

6. CONVENTIONS 10
   6.1. Symbols (and abbreviated terms) 10
   6.2. Identifiers 10
   6.3. Abbreviated terms 11
   6.4. Finding requirements and recommendations 11

7. REQUIREMENTS CLASS: PART 1 UML 13
   7.1. The ModSpec and the “Form” of a standard 13
   7.2. Optional Requirements class: UML model extension to the core 13

8. CONFORMANCE TEST CLASS: UML MODEL EXTENDS THE STANDARD 19
   8.1. Dependency on Core 19

8.2. The UML model is normative 19
8.3. Dependency graph must be explicit 19
8.4. Leaf packages in only one requirements class 20
8.5. Requirements class associated to a unique package 20
8.6. A requirements class contains complete leaf packages 20
8.7. Classes common to all requirement classes are in the core 21
8.8. Package dependencies become requirements class extensions 21
8.9. Dependencies in packages are reflected in dependencies in requirements classes 22
8.10. Co-dependent packages are in the same requirements class 22
8.11. All classes are in leaf packages 22

LIST OF TABLES

Table 1 ix
Table 2 13
Table 3 14
Table 4 14
Table 5 15
Table 6 15
Table 7 15
Table 8 15
Table 9 16
Table 10 16
Table 11 16
Table 12 16
Table 13 17

I ABSTRACT

This OGC member developed and approved document, referred to as the `ModSpec: Part 2 - UML`, defines an optional requirements class for UML

The ModSpec: Core Standard 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 GitHub repository for this document (https://github.com/opengeospatial/ogc-modspec).

II KEYWORDS

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

ogcdoc, OGC document, ModSpec, UML

III PREFACE

The ModSpec specifies a formal structure for standards documents but does not supply specific content. This Part 2 specifies the optional UML Requirements Class.

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

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

IV SECURITY CONSIDERATIONS

No security considerations have been made for this Standard.

V SUBMITTING ORGANIZATIONS

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

- Carl Reed, Heazeltech, ImageMatters

VI SUBMITTERS

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

Name	Affiliation
Carl Reed	Carl Reed & Associates
John Herring, PhD	Oracle

VII REVISION HISTORY

This is the second normative version of this document.

VIII FUTURE WORK

Improvements to this document will be made based on implementation and changing technical requirements.

IX CONTRIBUTORS

Additional contributors to this Standard include the following:

Name	Organization
Carl Reed, PhD	Carl Reed
Chuck Heazel	Heazeltech
Jeff Yutzler	ImageMatters

X ACKNOWLEDGEMENTS

The following OGC Members were key contributors to Version 1.0 of the ModSpec

Table 1

Name	Organization
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

1 SCOPE

1 SCOPE

The ModSpec defines characteristics and structure for the specification of Standards that will encourage implementation by minimizing difficulty determining requirements, mimicking implementation structure and maximizing usability and interoperability. This Part 2 of the ModSpec defines the requirement class for using UML. The use case is if the organizing mechanism for the data model used in a standard is an object model, then the mapping from parts of the model to the requirements classes should follow the logical mechanism described in Part 2 of the ModSpec.

1.1. Introduction

1.1.1. Semantics

The ModSpec Core Standard defines requirements for conformance to the ModSpec, but testing for that conformance may depend on how the various forms and parts of a conformant standard are viewed. The ModSpec Part 2 Standard specifies how those views are to be defined as UML.

As suggested in Clause Requirement 8 of the ModSpec 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.

1.1.2. ModSpec document structure

Version 1.1 of the ModSpec is split into a Core standard and multiple Parts. These are:

- Part 1: Core: Contains all the core requirements and informational text that define the model and internal structure of a standard.
- Part 2: UML Model requirements
- Part 3: XML Model and Schematron requirements

2

CONFORMANCE

2 CONFORMANCE

Conformance to the ModSpec: Part 2 — UML Standard by technical implementation standards can be tested by inspection. The test suite is in [annex-A].

There is one conformance class for this document for:

1. Standards using UML to state requirements, extending the core.

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

- The core requirements which shall be met by all conformant standards.
- The various subclauses of Clause 7 list requirements partially derived from the core, but more specific to the conditions where the data model is expressed in UML. This requirements class is an extension of the Core presented in the ModSpec Core Standard Clause 7.

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.

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

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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.

NOTE: All terms and definitions in Clause 4 of the OGC ModSpec: Core are relevant and used in this Part.

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/

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/

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/

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”)
OCL	Object Constraint Language (part of UML)
OGC	Open Geospatial Consortium (http://www.opengeospatial.org/)
OMG	Object Management Group (http://www.omg.org/)
OOP	Object Oriented Programming
OOPL	OOP Language (such as C++ or Java)
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 Part 2 — UML Standard is stated in one and only one place, in a standard format, and 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 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

REQUIREMENTS CLASS: PART 1 UML

7 REQUIREMENTS CLASS: PART 1 UML

This clause defines the key concepts and requirements that represent Part 2 of the ModSpec.

7.1. The ModSpec and the “Form” of a standard

NOTE: For OGC Standards, the assumption is that documents are in a commonly used logical form (template).

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

The ModSpec defines a “requirement” of a standard as an atomic testable criterion. See the formal definition of requirement in [term-requirement]

7.2. Optional Requirements class: UML model extension to the core

If the organizing mechanism for the data model used in the standard is an object model, then the mapping from parts of the model to the requirements classes should follow the logical mechanism described here.

The terminology used here is common to all versions of the UML standard, and may be applied to any such version.

First, by the requirements above, the extension relationship of this conformance test class to the core will be made explicit.

Table 2

Requirements Class — UML extension to the core	
/req/core/data-representation	
Target	ModSpec Conformant UML Model
Dependency	OGC ModSpec Version 1.1 [OGC 25-003r1]

REQ001	/req/part2/uml/conformance-with-core
REQ002	/req/part2/uml/object-model
REQ003	/req/part2/uml/dependency-graph
REQ004	/req/part2/uml/leaf-package
REQ005	/req/part2/uml/class-package
REQ006	/req/part2/uml/to-leaf
REQ007	/req/part2/uml/common-req-classes
REQ008	/req/part2/uml/source-target
REQ009	/req/part2/uml/leaf-package-dependency
REQ010	/req/part2/uml/two-way-dependency
REQ011	/req/part2/uml/segregate-into-leaf-packages

Any conformant UML extension will comply with the ModSpec Core requirements class.

Table 3

REQ001	**/req/part2/uml/conformance-with-core** An implementation passing the UML conformance test class *SHALL* first pass the ModSpec core conformance test class

Assuming all legitimate direct package dependencies are included in the UML model, the conformance/requirements class associated to a package A will directly reference the conformance/requirements class associated to another package B, if and only if A is dependent on B. Indirect dependencies will be dealt with as the conformance classes cascade.

This clause uses UML terminology, but other object modeling languages and, if needed, the object code itself can be mapped to a UML model.

Table 4

REQ002	**/req/part2/uml/object-model** To be conformant to this UML requirements class, UML *SHALL* be used to express the object model, either as the core mechanism of the standard or as a normative adjunct to formally explain the standard in a model

Table 5

REQ003	**/req/part2/uml/dependency-graph** A UML model *SHALL* have an explicit dependency graph for the leaf packages and external packages used by the standard consistent with the way their classifiers use those of other packages

NOTE: External packages having predated the current version of the standard will not normally have dependencies to packages within the standard.

Other dependencies (indirect) will arise from the transitive closure of the above direct dependencies. That means if A depends on B, and B depends on C then A depends on C. Since these indirect dependencies will show up in the cascade of included conformance tests based solely on the direct dependencies, we can ignore them for effects on structure.

Since a package can consist solely of other packages, there is always the capability to define in UML a single package that will correspond to a particular requirements class and its associated conformance class.

Table 6

REQ004	**/req/part2/uml/leaf-model** A UML leaf package *SHALL* be associated directly to only one requirements class.

Table 7

REQ005	**/req/part2/uml/class-package** Each requirements class shall be associated to a unique package in the model and include either directly or by a dependency any requirement associated to any of its subpackages.

The class definitions are the “requirements” of a UML expressed standard. Therefore, the logical consequence of the above is to organize requirements classes based on leaf packages.

Table 8

REQ006	**/req/part2/uml/to-leaf** A requirements class *SHALL* be associated to some number of complete leaf packages and all classes and constraints in those packages.

If a requirement uses or refers to elements of more than one package, then one of the packages will be called the source of the requirement, and the other targets. The requirement with the same source package will always be associated with the same requirements module and/or class.

Table 9

REQ007	**/req/part2/uml/common-req-classe** Classes that are common to all requirements classes *SHALL* be in a package associated to the core requirements class.

This is actually a derived requirement and is repeated here for emphasis.

This dependency of requirements classes will be consistent with the usual mechanism for describing the source and target of dependencies between packages. By Clause [req-22] in the Core, only classes in the source requirements class will be affected by the requirement.

In UML, source and target dependency relations are defined in such a way that the source of the relation is dependent on the target of the relation.

Table 10

REQ008		**/req/part2/uml/source-target**
	A	In the UML model, if a "source" package is dependent on a "target" package then their requirements class *SHALL* be equal or
	B	The source package's class *SHALL* be an extension of the target package's class.

This means that the dependency graph of the UML packages parallels in some sense the extension diagram of the requirements/conformance classes. If all leaf packages of the model are moved into "requirements class packages" containing their corresponding modeling packages the model then satisfies the following recommendation:

Each requirements class in a conformant standard should be associated to one and only one UML package (which may contain sub-packages for a finer level of structure). If the core requirements class contains only recommendations, it may be an exception to this.

Table 11

REQ009	**/req/part2/uml/leaf-package-dependency** If one leaf package is dependent on another leaf package, then the requirements class of the first *SHALL* be the same or an extension of the requirements class of the second.

Table 12

REQ010	**/req/part2/uml/two-way-dependency** If two packages have a two-way dependency (a "co-dependency"), they *SHALL* be associated to the same requirements class.

For example, if two classes have a two-way navigable association, then these two classes must be (transitively) in the same conformance requirements class package.

The hierarchical structure of a UML model is based on UML classes, residing in UML packages. UML packages can then reside in larger UML packages. Although there is nothing to demand it, it is a common practice to move all classes down the hierarchy to leaf packages. Leaf packages are those that contain only classes (that is, contain no smaller subpackages). Classes can act as packages in the sense that a UML class can contain a locally defined class whose scope is the class itself. For our purposes, we will consider a class and its contained local classes to all be in the package of the original class.

Table 13

REQ011	**/req/part2/uml/segregate-into-leaf-packages** The UML model *SHALL* segregate all classes into leaf packages.

8

CONFORMANCE TEST CLASS: UML MODEL EXTENDS THE STANDARD

8 CONFORMANCE TEST CLASS: UML MODEL EXTENDS THE STANDARD

8.1. Dependency on Core

An implementation passing the UML conformance test class shall first pass the core conformance test class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 3
4. Test Type: Conformance.

8.2. The UML model is normative

To be conformant to this UML conformance class, UML shall be used to express the object model, either as the core mechanism of the standard or as a normative adjunct to formally explain the standard in a model.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 4
4. Test Type: Conformance.

8.3. Dependency graph must be explicit

A UML model shall have an explicit dependency graph for the leaf packages and external packages used by the standard consistent with the way their classifiers use those of other packages.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 5
4. Test Type: Conformance.

8.4. Leaf packages in only one requirements class

A UML leaf package shall be associated directly to only one requirements class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 6
4. Test Type: Conformance.

8.5. Requirements class associated to a unique package

Each requirements class shall be associated to a unique package in the model and include either directly or by a dependency any requirement associated to any of its subpackages.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 7
4. Test Type: Conformance.

8.6. A requirements class contains complete leaf packages

A requirements class shall be associated to some number of complete leaf packages and all classes and constraints in those packages.

1. Test Purpose: Verify that this requirement is satisfied.

2. Test Method: Inspect the document to verify the above.
3. Reference: Table 8
4. Test Type: Conformance.

8.7. Classes common to all requirement classes are in the core

Classes that are common to all requirements classes shall be in a package associated to the core conformance/requirements class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 9.
4. Test Type: Conformance.

8.8. Package dependencies become requirements class extensions

In the UML model, if a “source” package is dependent on a “target” package then their requirements class shall be equal or the source package’s class shall be an extension of the target package’s class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 10.
4. Test Type: Conformance.

8.9. Dependencies in packages are reflected in dependencies in requirements classes

If one leaf package is dependent on another leaf package, then the requirements class of the first shall be the same or an extension of the requirements class of the second.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 11.
4. Test Type: Conformance.

8.10. Co-dependent packages are in the same requirements class

If two packages have a two-way dependency (a “co-dependency”), they shall be associated to the same requirements class.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 12
4. Test Type: Conformance.

8.11. All classes are in leaf packages

The UML model shall segregate all classes into leaf packages.

1. Test Purpose: Verify that this requirement is satisfied.
2. Test Method: Inspect the document to verify the above.
3. Reference: Table 13
4. Test Type: Conformance.