
**Information technology — Object
Management Group Object Constraint
Language (OCL)**

*Technologies de l'information — Langage de contraintes orienté-objet
(OCL) de l'OMG*

**iTeh STANDARD PREVIEW
(standards.iteh.ai)**

[ISO/IEC 19507:2012](https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012)

[https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-
f53178fd9581/iso-iec-19507-2012](https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012)

iTeh STANDARD PREVIEW (standards.iteh.ai)

ISO/IEC 19507:2012

<https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Table of Contents

Foreword	ix
Introduction	x
1 Scope	1
2 Conformance	1
3 References	2
3.1 Normative References	2
3.2 Informative References	2
4 Terms and Definitions	3
5 Notational Conventions	3
6 Additional Information	3
6.1 Changes to Adopted OMG Specifications	3
6.2 Structure of the Specification	3
6.3 Acknowledgements	4
7 OCL Language Description	5
7.1 General	5
7.2 Why OCL?	5
7.2.1 Where to Use OCL	5
7.3 Introduction	6
7.3.1 Legend	6
7.3.2 Example Class Diagram	6
7.3.3 Character Set	7
7.4 Relation to the UML Metamodel	7
7.4.1 Self	7
7.4.2 Specifying the UML Context	7
7.4.3 Invariants	8
7.4.4 Pre- and Postconditions	8
7.4.5 Package Context	9
7.4.6 Operation Body Expression	9
7.4.7 Initial and Derived Values	9
7.4.8 Other Types of Expressions	10
7.5 Basic Values and Types	10
7.5.1 Types from the UML Model	11
7.5.2 Enumeration Types	11
7.5.3 Let Expressions	11
7.5.4 Additional operations/attributes through «definition» expressions	12
7.5.5 Type Conformance	12

7.5.6 Re-typing or Casting	13
7.5.7 Precedence Rules	14
7.5.8 Use of Infix Operators	14
7.5.9 Keywords	15
7.5.10 Comment	16
7.5.11 Invalid Values	16
7.6 Objects and Properties	16
7.6.1 Properties: Attributes	17
7.6.2 Properties: Operations	17
7.6.3 Properties: AssociationEnds and Navigation	18
7.6.4 Navigation to Association Classes	20
7.6.5 Navigation from Association Classes	21
7.6.6 Navigation through Qualified Associations	21
7.6.7 Using Pathnames for Packages	21
7.6.8 Accessing overridden properties of supertypes	22
7.6.9 Predefined properties on All Objects	22
7.6.10 Features on Classes Themselves	23
7.6.11 Collections	24
7.6.12 Collections of Collections	25
7.6.13 Collection Type Hierarchy and Type Conformance Rules	25
7.6.14 Previous Values in Postconditions	25
7.6.15 Tuples	26
7.7 Collection Operations	27
7.7.1 Select and Reject Operations	27
7.7.2 Collect Operation	28
7.7.3 ForAll Operation	29
7.7.4 Exists Operation	30
7.7.5 Closure Operation	30
7.7.6 Iterate Operation	31
7.8 Messages in OCL	32
7.8.1 Calling operations and sending signals	32
7.8.2 Accessing result values	33
7.8.3 An example	33
7.9 Resolving Properties	34
8 Abstract Syntax	35
8.1 Introduction	35
8.2 The Types Package	35
8.2.1 Type Conformance	38
8.2.2 Operations and Well-formedness Rules for the Types Package	40
8.3 The Expressions Package	42
8.3.1 Expressions Core	43
8.3.2 FeatureCall Expressions	45
8.3.3 If Expressions	47
8.3.4 Message Expressions	48
8.4 Literal Expressions	49
8.4.1 Let Expressions	52
8.4.2 Well-formedness Rules of the Expressions package	53
8.4.3 Additional Operations on UML metaclasses	60

8.4.4 Additional Operations on OCL Metaclasses	62
8.4.5 Overview of class hierarchy of OCL Abstract Syntax metamodel	64
9 Concrete Syntax	65
9.1 General	65
9.2 Structure of the Concrete Syntax	65
9.3 A Note to Tool Builders	67
9.3.1 Parsing	67
9.3.2 Visibility	67
9.4 Concrete Syntax	67
9.4.1 ExpressionInOclCS	68
9.4.2 OclExpressionCS	68
9.4.3 VariableExpCS	69
9.4.4 simpleNameCS	69
9.4.5 restrictedKeywordCS	70
9.4.6 unreservedSimpleNameCS	71
9.4.7 pathNameCS	71
9.4.8 LiteralExpCS	72
9.4.9 EnumLiteralExpCS	72
9.4.10 CollectionLiteralExpCS	73
9.4.11 CollectionTypeIdentifierCS	73
9.4.12 CollectionLiteralPartsCS	74
9.4.13 CollectionLiteralPartCS	74
9.4.14 CollectionRangeCS	74
9.4.15 PrimitiveLiteralExpCS	75
9.4.16 TupleLiteralExpCS	76
9.4.17 UnlimitedNaturalLiteralExpCS	76
9.4.18 IntegerLiteralExpCS	76
9.4.19 RealLiteralExpCS	77
9.4.20 StringLiteralExpCS	77
9.4.21 BooleanLiteralExpCS	78
9.4.22 TypeLiteralExpCS	78
9.4.23 CallExpCS	79
9.4.24 LoopExpCS	79
9.4.25 IteratorExpCS	80
9.4.26 IterateExpCS	83
9.4.27 VariableDeclarationCS	84
9.4.28 TypeCS	85
9.4.29 primitiveTypeCS	85
9.4.30 oclTypeCS	86
9.4.31 collectionTypeCS	86
9.4.32 tupleTypeCS	87
9.4.33 variableDeclarationListCS	87
9.4.34 FeatureCallExpCS	88
9.4.35 OperationCallExpCS	88
9.4.36 PropertyCallExpCS	91
9.4.37 NavigationCallExpCS	93
9.4.38 AssociationClassCallExpCS	93
9.4.39 isMarkedPreCS	94
9.4.40 argumentsCS	94

9.4.41 LetExpCS	95
9.4.42 LetExpSubCS	95
9.4.43 OclMessageExpCS	96
9.4.44 OclMessageArgumentsCS	97
9.4.45 OclMessageArgCS	97
9.4.46 IfExpCS	98
9.4.47 NullLiteralExpCS	98
9.4.48 InvalidLiteralExpCS	99
9.4.49 Comments	99
9.5 Environment Definition	99
9.5.1 Environment	99
9.5.2 NamedElement	102
9.5.3 Namespace	102
9.6 Concrete to Abstract Syntax Mapping	102
9.7 Abstract Syntax to Concrete Syntax Mapping	103
10 Semantics Described Using UML	105
10.1 Introduction	105
10.2 The Values Package	106
10.2.1 Definitions of Concepts for the Values Package	107
10.2.2 Well-formedness Rules for the Values Package	111
10.2.3 Additional Operations for the Values Package	113
10.2.4 Overview of the Values Package	114
10.3 The Evaluations Package	115
10.3.1 Definitions of Concepts for the Evaluations Package	116
10.3.2 Well-formedness Rules of the Evaluations Package	125
10.3.3 Additional Operations of the Evaluations Package	132
10.3.4 Overview of the Values Package	133
10.4 The AS-Domain-Mapping Package	134
10.4.1 Well-formedness rules for the AS-Domain-Mapping.type-value Package	135
10.4.2 Additional Operations for the AS-Domain-Mapping.type-value Package	137
10.4.3 Well-formedness rules for the AS-Domain-Mapping.exp-eval Package	137
11 OCL Standard Library	145
11.1 Introduction	145
11.2 The OclAny, OclVoid, OclInvalid, and OclMessage Types	146
11.2.1 OclAny	146
11.2.2 OclMessage	146
11.2.3 OclVoid	146
11.2.4 OclInvalid	146
11.3 Operations and Well-formedness Rules	146
11.3.1 OclAny	146
11.3.2 OclVoid	148
11.3.3 OclMessage	148
11.4 Primitive Types	148
11.4.1 Real	148
11.4.2 Integer	148
11.4.3 String	149

11.4.4 Boolean	149
11.4.5 UnlimitedNatural	149
11.5 Operations and Well-formedness Rules	149
11.5.1 Real	149
11.5.2 Integer	150
11.5.3 String	151
11.5.4 Boolean	153
11.5.5 UnlimitedNatural	154
11.6 Collection-Related Types	155
11.6.1 Collection	156
11.6.2 Set	156
11.6.3 OrderedSet	156
11.6.4 Bag	156
11.6.5 Sequence	156
11.7 Operations and Well-formedness Rules	156
11.7.1 Collection	156
11.7.2 Set	159
11.7.3 OrderedSet	161
11.7.4 Bag	163
11.7.5 Sequence	165
11.8 Predefined Iterator Expressions	168
11.8.1 Extending the Standard Library with Iterator Expressions	168
11.9 Mapping Rules for Predefined Iterator Expressions	168
11.9.1 Collection	168
11.9.2 Set	170
11.9.3 Bag	171
11.9.4 Sequence	172
11.9.5 OrderedSet	173
12 The Use of OCL Expressions in UML Models	175
12.1 Introduction	175
12.2 The ExpressionInOcl Type	175
12.2.1 ExpressionInOcl	176
12.3 Well-formedness Rules	176
12.3.1 ExpressionInOcl	176
12.4 Standard Placements of OCL Expressions	177
12.4.1 How to Extend the Use of OCL at Other Places	177
12.5 Definition	177
12.5.1 Well-formedness Rules	177
12.6 Invariant	178
12.6.1 Well-formedness rules	178
12.7 Precondition	179
12.7.1 Well-formedness rules	179
12.7.2 Postcondition	180
12.7.3 Well-formedness rules	180
12.8 Initial Value Expression	180
12.8.1 Well-formedness rules	181
12.9 Derived Value Expression	182

12.10	Operation Body Expression	182
12.11	Guard	182
12.11.1	Well-formedness rules	183
12.12	Concrete Syntax of Context Declarations	183
12.12.1	packageDeclarationCS	184
12.12.2	contextDeclarationCS	184
12.12.3	propertyContextDeclCS	184
12.12.4	initOrDerValueCS	184
12.12.5	classifierContextDeclCS	184
12.12.6	invOrDefCS	184
12.12.7	defExpressionCS	185
12.12.8	operationContextDeclCS	185
12.12.9	prePostOrBodyDeclCS	185
12.12.10	operationCS	185
12.12.11	parametersCS	185
13	The Basic OCL and Essential OCL	187
13.1	Introduction	187
13.2	OCL Adaptation for Metamodeling	187
13.3	Diagrams	188
Annex A	Semantics (standards.iteh.ai)	193
Annex B	Bibliography	229
Annex C	Legal Information	231

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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

ISO/IEC 19507 was prepared by Technical Committee ISO/IEC JTC1, Information technology, in collaboration with the Object Management Group (OMG), following the submission and processing as a Publicly Available Specification (PAS) of the OMG Object Constraint Language specification Version 2.3.1.

ISO/IEC 19507, under the general title *Information technology - Open distributed processing - Object Constraint Language specification (OCL)*, apart from this introductory material is identical with that for the OMG specification for Object Constraint Language, v2.3.1.

Introduction

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability, and portability can be integrated.

RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. The scopes and objectives of the RM-ODP Part 2 and the UML, while related, are not the same and, in a number of cases, the RM-ODP Part 2 and the UML specification use the same term for concepts that are related but not identical (e.g., interface). Nevertheless, a specification using the Part 2 modeling concepts can be expressed using UML with appropriate extensions (using stereotypes, tags, and constraints).

RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2. Given the relation between UML as a modeling language and Part 2 of the RM ODP standard, it is easy to show that UML is suitable as a notation for the individual viewpoint specifications defined by the RM-ODP.

OCL Language

OCL is a pure specification language; therefore, an OCL expression is guaranteed to be without side effects. When an OCL expression is evaluated, it simply returns a value. It cannot change anything in the model. This means that the state of the system will never change because of the evaluation of an OCL expression, even though an OCL expression can be used to *specify* a state change (e.g., in a post-condition).

OCL is not a programming language; therefore, it is not possible to write program logic or flow control in OCL. You cannot invoke processes or activate non-query operations within OCL. Because OCL is a modeling language in the first place, OCL expressions are not by definition directly executable.

OCL is a typed language so that each OCL expression has a type. To be well formed, an OCL expression must conform to the type conformance rules of the language. For example, you cannot compare an Integer with a String. Each Classifier defined within a UML model represents a distinct OCL type. In addition, OCL includes a set of supplementary predefined types. These are described in Clause 11 (“The OCL Standard Library”).

Information technology - Object Management Group Object Constraint Language (OCL)

1 Scope

This International Standard defines the Object Constraint Language (OCL), version 2.3.1. OCL version 2.3.1 is the version of OCL that is aligned with UML 2.3 and MOF 2.0.

2 Conformance

The UML 2.0 Infrastructure and the MOF 2.0 Core specifications that were developed in parallel with this OCL 2.3.1 specification share a common core. The OCL specification contains a well-defined and named subset of OCL that is defined purely based on the common core of UML and MOF. This allows this subset of OCL to be used with both the MOF and the UML, while the full specification can be used with the UML only.

The following compliance points are distinguished for both parts.

1. Syntax compliance: The tool can read and write OCL expressions in accordance with the grammar, including validating its type conformance and conformance of well-formedness rules against a model.
2. XMI compliance: The tool can exchange OCL expressions using XMI.
3. Evaluation compliance: The tool evaluates OCL expressions in accordance with the semantics clause. The following additional compliance points are optional for OCL evaluators, as they are dependent on the technical platform on which they are evaluated:
 - allInstances()
 - pre-values and oclIsNew() in postconditions
 - OclMessage
 - navigating across non-navigable associations
 - accessing private and protected features of an object

The following table shows the possible compliance points. Each tool is expected to fill in this table to specify which compliance points are supported.

Table 2.1 - Overview of OCL Compliance Points

	OCL-MOF subset	Full OCL
Syntax		
XMI		
Evaluation		
- allInstances		
- @pre in postconditions		
- OclMessage		
- navigating non-navigable associations		
- accessing private and protected features		

3 References

iTech STANDARD PREVIEW
(standards.iteh.ai)

3.1 Normative References

The following referenced documents are indispensable for the application 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 19507:2012
<https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012>

- ISO 639 Codes for the representation of names of languages
- ISO 3166 Codes for the representation of names of countries and their subdivisions
- ISO/IEC 10646:2003 Information technology -- Universal Multiple-Octet Coded Character Set (UCS)
- UML 2.3 Superstructure Specification: <http://www.omg.org/spec/UML/2.3/Superstructure/PDF/>
- UML 2.3 Infrastructure Specification: <http://www.omg.org/soec/UML/2.3/Infrastructure/PDF/>
- MOF 2.0 Core Specification: <http://www.omg.org/spec/MOF/2.0/PDF/>
- UNICODE 5.1 Standard: <http://www.unicode.org/versions/Unicode5.1.0/>
- Unicode Technical Standard#10: <http://www.unicode.org/reports/tr10/>

3.2 Informative References

The following specifications are referenced in informative text:

- ISO/IEC 19501:2005 Information technology - Open Distributed Processing -- Unified Modeling Language (UML) Version 1.4.2

4 Terms and Definitions

There are no formal definitions that are taken from other documents.

5 Notational Conventions

There are no symbols defined.

6 Additional Information

6.1 Changes to Adopted OMG Specifications

This International Standard replaces the specification of OCL given in OCL 2.2.

The version of OCL specified in ISO/IEC 19501:2005 is intended for use in models based on UML 1.4.1 and UML 1.5. However, use of the OCL specified by ISO/IEC 19501:2005 is not prescribed by this specification.

The version of OCL specified in this International Standard is not directly applicable to models based on ISO/IEC 19501:2005.

6.2 Structure of the Specification

This International Standard is divided into several clauses.

- The OCL Language Description clause gives an informal description of OCL. This clause is not normative, but meant to be explanatory.
- Clause 8 (“Abstract Syntax”) describes the abstract syntax of OCL using a MOF 2.0 compliant metamodel. This is the same approach as used in the UML specifications. The metamodel is MOF compliant in the sense that it only uses constructs that are defined in the MOF.
- Clause 9 (“Concrete Syntax”) describes the canonical concrete syntax using an attributed EBNF grammar. This syntax is mapped onto the abstract syntax, achieving a complete separation between concrete and abstract syntax.
- Clause 10 (“Semantics Described using UML”) describes the semantics for OCL using UML.
- In Clause 11 (“The OCL Standard Library”) the OCL Standard Library is described. This defines type like Integer, Boolean, etc. and all the collection types. OCL is not a stand-alone language, but an integral part of the UML. An OCL expression needs to be placed within the context of a UML model.
- Clause 12 (“The Use of Ocl Expressions in UML Models”) describes a number of places within the UML where OCL expressions can be used.
- Clause 13 (“Basic OCL and Essential OCL”) defines the adaptation of the OCL metamodel when used in particular context of Core::Basic infrastructure library package and in the context of EMOF.
- Annex A (Semantics), Annex B (Bibliography), and Annex C (Legal Information)

6.3 Acknowledgements

The following companies submitted and/or supported parts of this specification:

- Adaptive Ltd.
- BoldSoft
- Borland Software Corporation
- Compuware Corporation
- Dresden University of Technology
- France Telecom
- International Business Machines
- IONA
- Kabira Technologies Inc.
- Kings College
- Classe Objecten
- Open Canarias, SL
- Oracle
- Project Technology Inc.
- Rational Software Corporation
- SAP AG
- Softeam
- Syntropy Ltd.
- Telelogic
- Thales
- University of Bremen
- University of Kent
- University of York
- Zeligsoft, Inc.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 19507:2012](https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012)

<https://standards.iteh.ai/catalog/standards/sist/042fd345-859c-4fc9-b043-f53178fd9581/iso-iec-19507-2012>

7 OCL Language Description

7.1 General

This clause introduces the Object Constraint Language (OCL), a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model. Note that when the OCL expressions are evaluated, they do not have side effects (i.e., their evaluation cannot alter the state of the corresponding executing system).

OCL expressions can be used to specify operations / actions that, when executed, do alter the state of the system. UML modelers can use OCL to specify application-specific constraints in their models. UML modelers can also use OCL to specify queries on the UML model, which are completely programming language independent.

Note - This clause is informative only and not normative.

7.2 Why OCL?

A UML diagram, such as a class diagram, is typically not refined enough to provide all the relevant aspects of a specification. There is, among other things, a need to describe additional constraints about the objects in the model. Such constraints are often described in natural language. Practice has shown that this will always result in ambiguities. In order to write unambiguous constraints, so-called formal languages have been developed. The disadvantage of traditional formal languages is that they are usable to persons with a strong mathematical background, but difficult for the average business or system modeler to use.

OCL has been developed to fill this gap. It is a formal language that remains easy to read and write. It has been developed as a business modeling language within the IBM Insurance division, and has its roots in the Syntropy method.

OCL is a pure specification language; therefore, an OCL expression is guaranteed to be without side effects. When an OCL expression is evaluated, it simply returns a value. It cannot change anything in the model. This means that the state of the system will never change because of the evaluation of an OCL expression, even though an OCL expression can be used to *specify* a state change (e.g., in a post-condition).

OCL is not a programming language; therefore, it is not possible to write program logic or flow control in OCL. You cannot invoke processes or activate non-query operations within OCL. Because OCL is a modeling language in the first place, OCL expressions are not by definition directly executable.

OCL is a typed language so that each OCL expression has a type. To be well formed, an OCL expression must conform to the type conformance rules of the language. For example, you cannot compare an Integer with a String. Each Classifier defined within a UML model represents a distinct OCL type. In addition, OCL includes a set of supplementary predefined types. These are described in Clause 11 (“The OCL Standard Library”).

As a specification language, all implementation issues are out of scope and cannot be expressed in OCL.

The evaluation of an OCL expression is instantaneous. This means that the states of objects in a model cannot change during evaluation.

7.2.1 Where to Use OCL

OCL can be used for a number of different purposes:

- as a query language,