
UML Profile for Metaobject Facility (MOF) Specification

February 2004
Version 1.0
formal/04-02-06



OBJECT MANAGEMENT GROUP

An Adopted Specification of the Object Management Group, Inc.

Copyright © 2000, 2001, CBOP
Copyright © 2000, 2001, Data Access Technologies
Copyright © 2000, 2001, DSTC
Copyright © 2000, 2001, EDS
Copyright © 2000, 2001, Fujitsu
Copyright © 2000, 2001, IBM
Copyright © 2000, 2001, Iona Technologies
Copyright © 2003, Object Management Group, Inc.
Copyright © 2000, 2001, Open_IT
Copyright © 2000, 2001, Sun Microsystems
Copyright © 2000, 2001, Unisys Corporation

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

The OMG Object Management Group Logo®, CORBA®, CORBA Academy®, The Information Brokerage®, XMI® and IOP® are registered trademarks of the Object Management Group. OMG™, Object Management Group™, CORBA logos™, OMG Interface Definition Language (IDL)™, The Architecture of Choice for a Changing World™, CORBA services™, CORBA facilities™, CORBA med™, CORBA net™, Integrate 2002™, Middleware That's Everywhere™, UML™, Unified Modeling Language™, The UML Cube logo™, MOF™, CWM™, The CWM Logo™, Model Driven Architecture™, Model Driven Architecture Logos™, MDA™, OMG Model Driven Architecture™, OMG MDA™ and the XMI Logo™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

ISSUE REPORTING

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page <http://www.omg.org>, under Documents & Specifications, Report a Bug/Issue.

Contents

Preface	v
1. Introduction	1-1
1.1 Guide to the Specification	1-1
1.1.1 Overall Structure of the Specification	1-1
1.2 Conformance Issues	1-1
1.2.1 Compliance Points	1-1
2. UML Profile for MOF	2-3
2.1 Introduction	2-4
2.2 Mapping Table	2-5
2.3 ModelElement	2-6
2.3.1 Tags on UML ModelElement	2-6
2.3.2 ModelElement Property Map	2-6
2.3.3 ModelElement Constraints	2-6
2.3.4 ModelElement Limitations	2-6
2.4 Package	2-7
2.4.1 Tags on UML Model with Stereotype <<metamodel>>	2-7
2.4.2 Model-to-Package Property Map	2-7
2.4.3 Model-to-Package Constraints	2-7
2.4.4 Model-to-Package Limitations	2-8
2.5 Import	2-8
2.5.1 Tags on UML ElementImport	2-8
2.5.2 ElementImport-to-Import Property Map	2-8
2.5.3 ElementImport-to-Import Constraints	2-8

Contents

2.5.4	ElementImport-to-Import Limitations	2-8
2.6	Class.	2-8
2.6.1	Tags on UML Class	2-9
2.6.2	Class Property Map	2-9
2.6.3	Class Constraints	2-9
2.6.4	Class Limitations	2-9
2.7	Attribute.	2-9
2.7.1	Tags on UML Attribute with No Stereotype.	2-9
2.7.2	Attribute Property Map.	2-10
2.7.3	Attribute Constraints	2-10
2.7.4	Attribute Limitations	2-10
2.8	Reference	2-10
2.8.1	Tags on UML Attribute with Stereotype <<reference>>	2-11
2.8.2	Explicit Reference Property Map	2-11
2.8.3	Implicit Reference Property Map	2-11
2.8.4	Reference Constraints.	2-11
2.8.5	Reference Limitations	2-12
2.9	Operation	2-12
2.9.1	Tags on UML Operation.	2-12
2.9.2	Operation Property Map.	2-12
2.9.3	Operation Constraints.	2-12
2.9.4	Operation Limitations.	2-12
2.10	Parameter	2-13
2.10.1	Tags on UML Parameter.	2-13
2.10.2	Parameter Property Map.	2-13
2.10.3	Parameter Constraints.	2-13
2.10.4	Parameter Limitations	2-13
2.11	Exception	2-13
2.11.1	Tags on UML Exception.	2-13
2.11.2	Exception Property Map.	2-14
2.11.3	Exception Constraints.	2-14
2.11.4	Exception Limitations	2-14
2.12	Exception Parameter	2-14
2.12.1	Tags on Attribute of UML Exception	2-14
2.12.2	Attribute-to-Parameter Property Map.	2-14
2.12.3	Attribute-to-Parameter Constraints.	2-14
2.12.4	Attribute-to-Parameter Limitations.	2-14
2.13	Association.	2-15
2.13.1	Tags on UML Association	2-15

2.13.2	Association Property Map	2-15
2.13.3	Association Constraints	2-15
2.13.4	Association Limitations	2-15
2.14	AssociationEnd	2-15
2.14.1	Tags on UML AssociationEnd	2-15
2.14.2	AssociationEnd Property Map	2-16
2.14.3	AssociationEnd Constraints	2-16
2.14.4	AssociationEnd Limitations	2-16
2.15	DataType	2-16
2.15.1	Tags on UML DataType	2-17
2.15.2	DataType Property Map	2-17
2.15.3	DataType Constraints	2-17
2.15.4	DataType Limitations	2-17
2.16	Constant	2-18
2.16.1	Tags on UML DataValue	2-18
2.16.2	DataValue-to-Constant Property Map	2-18
2.16.3	DataValue-to-Constant Constraints	2-18
2.16.4	DataValue-to-Constant Limitations	2-18
2.17	Constraint	2-18
2.17.1	Tags on UML Constraint	2-18
2.17.2	Constraint Property Map	2-19
2.17.3	Constraint Constraints	2-19
2.17.4	Constraint Limitations	2-19
2.18	Generalizes	2-19
2.18.1	Tags on UML Generalization	2-19
2.18.2	Generalization-to-Generalizes Property Map	2-19
2.18.3	Generalization-to-Generalizes Constraints	2-19
2.18.4	Generalization-to-Generalizes Limitations	2-19
2.19	Tag	2-19
2.19.1	Tags on UML TaggedValue	2-20
2.19.2	TaggedValue-to-Tag Property Map	2-20
2.19.3	TaggedValue-to-Tag Constraints	2-20
2.19.4	TaggedValue-to-Tag Limitations	2-20
2.20	Modularity	2-21
2.21	Associations	2-21
2.22	References	2-22
2.23	DataTypes	2-22
2.24	Names	2-22

Contents

Preface

About the Object Management Group

The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common framework for application development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heterogeneous environments. Conformance to these specifications will make it possible to develop a heterogeneous applications environment across all major hardware platforms and operating systems.

OMG's objectives are to foster the growth of object technology and influence its direction by establishing the Object Management Architecture (OMA). The OMA provides the conceptual infrastructure upon which all OMG specifications are based.

Intended Audience and Use

The information described in this manual is aimed at managers and software designers who want to produce applications that comply with the family of OMG standards. The benefit of compliance is, in general, to be able to produce interoperable applications that run in heterogeneous, distributed environments.

Context of OMG Modeling

The OMG is dedicated to producing a framework and specifications for commercially available object-oriented environments. The Object Management Architecture (as defined in the *Object Management Architecture Guide*) is the umbrella architecture for OMG specifications. The defining model for the architecture is the Reference Model,

which classifies the components, interfaces, and protocols that compose an object system. The Reference Model consists of the following components:

- **Object Request Broker**, which enables objects to transparently make and receive requests and responses in a distributed environment. It is the foundation for building applications from distributed objects and for interoperability between applications in hetero- and homogeneous environments. The architecture and specifications of the Object Request Broker are described in *CORBA: Common Object Request Broker Architecture and Specification*
- **Object Services**, a collection of services (interfaces and objects) that support basic functions for using and implementing objects. Services are necessary to construct any distributed application and are always independent of application domains. For example, the Life Cycle Service defines conventions for creating, deleting, copying, and moving objects; it does not dictate how the objects are implemented in an application. Specifications for Object Services are contained in *CORBA services: Common Object Services Specification*.
- **Common Facilities**, a collection of services that many applications may share, but which are not as fundamental as the Object Services. For instance, a system management or electronic mail facility could be classified as a common facility.
- **Application Objects**, which are objects specific to particular commercial products or end user systems. Application Objects correspond to the traditional notion of applications, so they are not standardized by the OMG. Instead, Application Objects constitute the uppermost layer of the Reference Model.
- **OMG Modeling**, a collection of modeling specifications that advance the state of the industry by enabling OO visual modeling tool interoperability. OMG Modeling provides a set of CORBA interfaces that can be used to define and manipulate a set of interoperable metamodels.

OMG formal documents are available from our web site in PostScript and PDF format. To obtain print-on-demand books in the documentation set or other OMG publications, contact the Object Management Group, Inc., at:

OMG Headquarters
250 First Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
pubs@omg.org
<http://www.omg.org>

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Helvetica bold - OMG Interface Definition Language (OMG IDL) and syntax elements.

Courier bold - Programming language elements.

Helvetica - Exceptions

Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Acknowledgments

This specification was prepared by the following companies:

- CBOP
- Data Access Technologies
- DSTC
- EDS
- Fujitsu
- IBM
- Iona Technologies
- Open-IT
- Sun Microsystems
- Unisys

Supporting companies are:

- Adaptive
- Hitachi
- Netaccount
- SINTEF

1.1 Guide to the Specification

1.1.1 Overall Structure of the Specification

Chapter 1 introduces the specification.

Chapter 2 contains the UML Profile for MOF, which is a normative two way mapping between UML and the MOF. This has been deemed essential, since, for the profiles to be understood, it is necessary to include metamodels that explain the concepts that the profiles express.

1.2 Conformance Issues

1.2.1 Compliance Points

UML Profile for MOF

Any tool that implements the Profile mechanisms defined in UML 1.4 (ad/01-02-13), and which is populated with stereotypes, tagged values and constraints defined in the uml2mof «profile» Package (Chapter 2).

Contents

This chapter includes the following topics.

Topic	Page
<i>Section I - Introduction</i>	
“Introduction”	2-2
<i>Section II - UML to MOF Mapping Table</i>	
“Mapping Table”	2-3
<i>Section III - Mapping Details</i>	
“ModelElement”	2-4
“Package”	2-5
“Import”	2-6
“Class”	2-6
“Attribute”	2-7
“Reference”	2-8
“Operation”	2-10
“Parameter”	2-11
“Exception”	2-11
“Exception Parameter”	2-12
“Association”	2-13
“AssociationEnd”	2-13

Topic	Page
“DataType”	2-14
“Constant”	2-16
“Constraint”	2-16
“Generalizes”	2-17
“Tag”	2-17
<i>Section IV - Guidelines</i>	
“Modularity”	2-19
“Associations”	2-19
“References”	2-20
“DataTypes”	2-20
“Names”	2-20

Section I - Introduction

2.1 Introduction

This chapter describes a mapping between the Unified Modeling Language (UML) and the Metaobject Facility (MOF). The two-way mapping supports both designing metamodels with UML (UML to MOF) and viewing metamodels with UML (MOF to UML). The sections in this chapter provide a table showing the mapping of element types, detailed mapping descriptions for individual element types, and guidelines for designing metamodels using UML.

The mapping is a UML profile. Per the definition of a UML profile, this chapter contains the following information.

UML Profile Requirements	Where Requirements are Satisfied in this Chapter
UML metamodel elements supported by the profile.	All metamodel elements are listed in the UML-to-MOF Mapping Table below.
Features defined by the profile as new metamodel elements.	This profile defines no new metamodel elements.
Common model elements predefined by the profile.	Stereotypes are listed in the UML-to-MOF Mapping Table below. There are no other predefined elements.

Features defined by the profile using standard extension mechanisms.	UML stereotypes are listed in the UML-to-MOF Mapping Table below. A mapping section for each element type includes a subsection identifying UML tags used by the profile.
Natural language prose that informally defines the semantics of the profile.	The body of this chapter explains the UML-to-MOF mapping — separate sections explain each element type.
Well-formedness rules that formally define the semantics of the profile.	A mapping section for each element type below includes a subsection expressing precisely how properties are mapped and a subsection listing constraints.

The profile has limitations. Some MOF details cannot be rendered in UML using this profile. The mapping section for each element type includes a subsection listing specific limitations.

Section II - UML-to-MOF Mapping Table

2.2 Mapping Table

The following UML elements and stereotypes are supported by the profile. Each maps to a specific MOF element as shown in the table below

UML Element	Stereotype	MOF Element
Model	<<metamodel>>	Package
ElementImport		Import
Class		Class
Attribute		Attribute
Attribute	<<reference>>	Reference
Operation		Operation
Parameter		Parameter
Exception		Exception
Attribute (within an Exception)		Parameter
Association		Association
AssociationEnd		AssociationEnd
DataType		DataType
DataValue		Constant
Constraint		Constraint
Generalization		Generalizes
TaggedValue		Tag

Section III - Mapping Details

The profile applies to an entire UML Model stereotyped as a <<metamodel>>. The profile applies to all elements contained directly or indirectly by the Model through composite associations. Hence, stereotypes are not generally needed for contained elements. All contained elements must be supported by the profile.

Separate sections below explain the mappings for each element type. Each section contains subsections covering these topics: tags, mapping properties, constraints, and limitations.

Tags are used for MOF properties not directly supported by UML. Except for the standard UML tag “documentation,” all tags used by the profile are prefixed with “org.omg.uml2mof” to mark them as belonging to this profile. All tags are optional unless specified as required.

2.3 ModelElement

In both UML and MOF, ModelElement is an abstract class. General tags and constraints on ModelElements are described below. The property map described below applies to the general cases where a UML ModelElement maps to a MOF ModelElement. In some cases the general map for a property is overridden for specific subclasses of MOF ModelElement.

2.3.1 Tags on UML ModelElement

Tag	Value
documentation	an annotation for the ModelElement

2.3.2 ModelElement Property Map

MOF Property	UML Property or Value
name	name
annotation	value of taggedValue with tag = “documentation”; otherwise “”
container	namespace
constraints	constraint

2.3.3 ModelElement Constraints

All constraints imposed by the MOF Specification are implicitly imposed based on the mapping to MOF defined herein.

Every UML ModelElement that maps to an MOF ModelElement must have a name.

2.3.4 ModelElement Limitations

None.

2.4 Package

A UML Model stereotyped as a <<metamodel>> maps to a MOF Package. Within a UML Model that maps to a MOF Package, any nested Model also maps to a MOF Package.

2.4.1 Tags on UML Model with Stereotype <<metamodel>>

Tag	Value
org.omg.uml2mof.clusteredImport	Comma-separated list of names of MOF Import objects that are clustered.
org.omg.uml2mof.hasImplicitReferences	“false” to prevent MOF References from being implied by AssociationEnds; “true” or no tag to imply a MOF Reference for each navigable AssociationEnd whose association and opposite end’s type are owned by the same package.

2.4.2 Model-to-Package Property Map

MOF Property	UML Property or Value
container	If namespace is a UML Model that is mapped to a MOF Package by this profile, then the package. If namespace is null or is not mapped to a MOF Package, then null.
contents	ownedElement, taggedValue*
isAbstract	isAbstract
isRoot	isRoot
isLeaf	isLeaf
supertypes	generalization.parent

* See section on MOF Tag about which tags are mapped to MOF Package contents

2.4.3 Model-to-Package Constraints

All UML elements contained by the Model through composite associations transitively are limited to the types and stereotypes named in this profile.

All constraints imposed by the MOF Specification are implicitly imposed, based on the mapping to MOF defined herein, on the UML Model and all of its contents.

All names listed for a tag of “org.omg.uml2mof.clusteredImport” must match names of MOF Import objects in the contents of the MOF Package.

A UML Model representing a nested MOF Package must not have a tag of “org.omg.uml2mof.hasImplicitReferences.”

UML ownedElement must be ordered.

UML taggedValue must be ordered.

2.4.4 Model-to-Package Limitations

The order of MOF Package.contents are not fully preserved when rendered using the profile because UML has separate associations for ownedElement and taggedValue.

2.5 Import

A UML ElementImport maps directly to a MOF Import.

2.5.1 Tags on UML ElementImport

None. Tags are not supported because UML ElementImport is not a ModelElement.

2.5.2 ElementImport-to-Import Property Map

MOF Property	UML Property or Value
name	alias if given, otherwise importedElement.name
annotation	none
container	package
visibility	visibility
isClustered	If package has a taggedValue with tag = "org.omg.uml2mof.clusteredImport" and the value includes the name of the MOF Import, then true; otherwise false
imported	importedElement

2.5.3 ElementImport-to-Import Constraints

The importedElement must be either a UML Model stereotyped as a <<metamodel>> or a Class owned directly or indirectly within such a Model.

2.5.4 ElementImport-to-Import Limitations

The profile does not support annotation of an Import.

2.6 Class

A UML Class maps directly to a MOF Class.

2.6.1 Tags on UML Class

Tag	Value
org.omg.uml2mof.isSingleton	“true” or “false” indicating a value for isSingleton

2.6.2 Class Property Map

MOF Property	UML Property or Value
contents	ownedElement followed by feature (in order)
visibility	visibility
isAbstract	isAbstract
isRoot	isRoot
isLeaf	isLeaf
supertypes	generalization.parent
isSingleton	value of taggedValue with tag = “org.omg.uml2mof.isSingleton”; otherwise false

2.6.3 Class Constraints

UML ownedElement must be ordered.

2.6.4 Class Limitations

The order of MOF Class.contents are not fully preserved when rendered using the profile because UML has separate associations for ownedElement and feature.

2.7 Attribute

A UML Attribute with no stereotype maps to a MOF Attribute.

2.7.1 Tags on UML Attribute with No Stereotype

Tag	Value
org.omg.uml2mof.isUnique	“true” or “false” indicating a value for isUnique
org.omg.uml2mof.isOrdered	“true” or “false” indicating a value for isOrdered
org.omg.uml2mof.isDerived	“true” or “false” indicating a value for isDerived

2.7.2 Attribute Property Map

MOF Property	UML Property or Value
container	owner
visibility	visibility
scope	ownerScope
type	type
multiplicity	multiplicity.range; isUnique and isOrdered are false unless specified with tags shown above
isChangeable	changeability = changeable
isDerived	Value of taggedValue with tag = "org.omg.uml2mof.isDerived" otherwise false.

2.7.3 Attribute Constraints

UML changeability must be either changeable or frozen.

UML multiplicity must have a single range.

2.7.4 Attribute Limitations

None.

2.8 Reference

A UML Attribute stereotyped as a <<reference>> maps to a MOF Reference.

Also, if the UML Model representing the outermost containing MOF Package does not have a tag of "org.omg.uml2mof.hasImplicitReferences" with a value of "false," then a MOF Reference is implied by each eligible UML AssociationEnd. An end is considered eligible if it is navigable, there is no explicit MOF Reference for that end within the same outermost MOF Package, and the end's association is owned by the same package that owns its opposite end's type (so as to not create circular package dependencies).

2.8.1 Tags on UML Attribute with Stereotype <<reference>>

Tag	Value
org.omg.uml2mof.referencedEnd	The name of an opposite AssociationEnd that is the referencedEnd.

2.8.2 Explicit Reference Property Map

MOF Property	UML Property or Value
container	owner
visibility	visibility
scope	ownerScope
type	type
multiplicity	multiplicity.range; isUnique and isOrdered are taken from the referencedEnd
isChangeable	changeability = changeable
referencedEnd	The one AssociationEnd of owner.allOppositeAssociationEnds that is identified by a taggedValue on the UML Attribute with tag = "org.omg.uml2mof.referencedEnd," or lacking a taggedValue, the UML Attribute's name.

2.8.3 Implicit Reference Property Map

MOF Property	UML Property or Value
name	referencedEnd's name
annotation	««»
container	The type of the AssociationEnd opposite to the referencedEnd.
constraints	none
visibility	referencedEnd's visibility
scope	instance_level
type	referencedEnd's type
multiplicity	referencedEnd's multiplicity
isChangeable	referencedEnd's isChangeable
referencedEnd	The AssociationEnd that implies the Reference.

2.8.4 Reference Constraints

UML changeability must be either changeable or frozen.

UML multiplicity must have a single range.

For a UML Attribute with a <<reference>> stereotype, if there is a UML taggedValue with tag = “org.omg.uml2mof.referencedEnd,” it must identify a visible AssociationEnd from among the Attribute’s owner.allOppositeAssociationEnds. If no such taggedValue is present, the UML Attribute name must identify a visible AssociationEnd from among the Attribute’s owner.allOppositeAssociationEnds.

For a UML Attribute with a <<reference>> stereotype, if the Attribute’s name is also the name of an AssociationEnd from among the Attribute’s owner.allOppositeAssociationEnds, then the Attribute makes explicit the pseudo-attribute implied by the name of the AssociationEnd. The Attribute’s name does not conflict with the pseudo-attribute name. Rather, the Attribute makes the pseudo-attribute explicit in the class. In this case, the Attribute must not have a taggedValue identifying a different AssociationEnd than the one identified by the Attribute’s name.

2.8.5 Reference Limitations

None.

2.9 Operation

A UML Operation maps to a MOF Operation.

2.9.1 Tags on UML Operation

None.

2.9.2 Operation Property Map

MOF Property	UML Property or Value
container	owner
contents	parameter
visibility	visibility
scope	ownerScope
isQuery	isQuery
exceptions	raisedSignal

2.9.3 Operation Constraints

UML raisedSignal must be ordered.

Each UML raisedSignal must be an Exception mapped by the profile.

2.9.4 Operation Limitations

Unlike a MOF Operation, a UML Operation cannot contain a Constraint. Therefore the profile does not support an Operation containing a Constraint.

2.10 Parameter

A UML Parameter maps to a MOF Parameter.

2.10.1 Tags on UML Parameter

Tag	Value
org.omg.uml2mof.multiplicity	a multiplicity range such as “0..1”, “*” or “1..*”
org.omg.uml2mof.isOrdered	“true” or “false” indicating a value for isOrdered
org.omg.uml2mof.isUnique	“true” or “false” indicating a value for isUnique

2.10.2 Parameter Property Map

MOF Property	UML Property or Value
container	behavioralFeature
type	type
direction	kind
multiplicity	Lower and upper are 1, and isOrdered and isUnique are false, unless specified with tags shown above.

2.10.3 Parameter Constraints

UML changeability must be either changeable or frozen.

A multiplicity specified by a taggedValue with tag = “org.omg.uml2mof.multiplicity” must represent a single valid multiplicity range.

2.10.4 Parameter Limitations

None.

2.11 Exception

A UML Exception maps to a MOF Exception. A UML Exception is a Signal, which is a Classifier, whereas MOF Exception is BehavioralFeature. For this reason, UML Attributes of an Exception, rather than UML Parameters, represent MOF Exception Parameters in the profile.

2.11.1 Tags on UML Exception

None.

2.11.2 Exception Property Map

MOF Property	UML Property or Value
contents	feature
visibility	visibility
scope	classifier

2.11.3 Exception Constraints

Each feature of the UML Exception must be an Attribute.

2.11.4 Exception Limitations

The profile does not support an Exception having instance-level scope.

2.12 Exception Parameter

An Attribute of a UML Exception maps to a Parameter of a MOF Exception.

2.12.1 Tags on Attribute of UML Exception

Tag	Value
org.omg.uml2mof.isOrdered	“true” or “false” indicating a value for isOrdered
org.omg.uml2mof.isUnique	“true” or “false” indicating a value for isUnique

2.12.2 Attribute-to-Parameter Property Map

MOF Property	UML Property or Value
container	owner
type	type
direction	out
multiplicity	Multiplicity.range; isOrdered and isUnique are false unless specified with tags shown above.

2.12.3 Attribute-to-Parameter Constraints

None.

2.12.4 Attribute-to-Parameter Limitations

None.

2.13 Association

A UML Association maps directly to a MOF Association.

A UML Association stereotyped as <<implicit>> is ignored by the profile and is not mapped to a MOF Association.

2.13.1 Tags on UML Association

None.

2.13.2 Association Property Map

MOF Property	UML Property or Value
contents	ownedElement, connection
visibility	visibility
isAbstract	isAbstract
isRoot	isRoot
isLeaf	isLeaf
supertypes	generalization.parent

2.13.3 Association Constraints

An Association must have exactly two ends.

2.13.4 Association Limitations

The order of MOF Class.contents are not fully preserved when rendered using the profile because UML has separate associations for ownedElement and connection.

2.14 AssociationEnd

A UML AssociationEnd maps directly to a MOF AssociationEnd.

2.14.1 Tags on UML AssociationEnd

None.

2.14.2 *AssociationEnd Property Map*

MOF Property	UML Property or Value
container	association
type	type
multiplicity	multiplicity.range, isUnique maps to upper > 1, isOrdered maps to ordering = ordered
aggregation	aggregation (UML aggregate matches MOF shared)
isNavigable	isNavigable
isChangeable	changeability = changeable

2.14.3 *AssociationEnd Constraints*

An Association must have exactly two ends.

UML changeability must be either changeable or frozen.

UML multiplicity must have a single range.

2.14.4 *AssociationEnd Limitations*

None.

2.15 *Data Type*

A UML Data Type maps directly to a MOF Data Type.

2.15.1 Tags on UML DataType

Tag	Value
org.omg.uml2mof.corbaType	CORBA IDL type name or type declaration.
org.omg.uml2mof.repositoryId	A repository id applicable within a typeCode constructed from a CORBA IDL type declaration.

2.15.2 DataType Property Map

MOF Property	UML Property or Value
contents	TypeAlias objects as required by taggedValue with tag = "org.omg.uml2mof.corbaType"*
visibility	visibility
isAbstract	isAbstract
isRoot	isRoot
isLeaf	isLeaf
supertypes	generalization.parent
typeCode	Value of taggedValue with tag = "org.omg.uml2mof.corbaType"*; otherwise, a typeCode based on name.**

* If a TaggedValue specifies a CORBA type, the value is parsed to determine the typeCode. If the value simply names a type, then it must name a CORBA primitive type. Otherwise, the value must be an IDL type declaration. Wherever the declaration refers by name to a Classifier contained in or imported into the metamodel, a MOF TypeAlias is constructed to reference the named Classifier. If there is a taggedValue with tag = "org.omg.uml2mof.repositoryId", then its value is used wherever a repository id can be specified within the typeCode.

** If a TaggedValue does not specify a CORBA type, then a CORBA type is determined from the name. The name matching is case-insensitive. If the name matches the name of a standard CORBA type, then that type is used. All other names revert to a typedef for the CORBA string type.

2.15.3 DataType Constraints

The value of a taggedValue with tag = "org.omg.uml2mof.corbaType" must identify a valid CORBA type.

UML ownedElement must be ordered.

2.15.4 DataType Limitations

A CORBA typecode contains information that is not revealed in an IDL rendering of a type. Such information is not handled by the profile.

The order of MOF DataType.contents are not fully preserved when rendered using the profile because TypeAlias objects are listed via a taggedValue separately from UML ownedElement.

2.16 Constant

A UML DataValue maps to a MOF Constant.

2.16.1 Tags on UML DataValue

Tag	Value
org.omg.uml2mof.constantValue	the value of the constant

2.16.2 DataValue-to-Constant Property Map

MOF Property	UML Property or Value
type	classifier
value	Value of taggedValue with tag = "org.omg.uml2mof.constantValue."

2.16.3 DataValue-to-Constant Constraints

A taggedValue is required to provide the Constant value. The value of the taggedValue must be a string representation of a valid value for the Constant's type.

2.16.4 DataValue-to-Constant Limitations

None.

2.17 Constraint

A UML Constraint maps directly to a MOF Constraint.

2.17.1 Tags on UML Constraint

Tag	Value
org.omg.uml2mof.evaluationPolicy	"immediate" or "deferred" indicating a value for evaluationPolicy.

2.17.2 Constraint Property Map

MOF Property	UML Property or Value
expression	body.body
language	body.language
evaluationPolicy	value from taggedValue with tag = "org.omg.uml2mof.evaluationPolicy"; otherwise deferred.
constrainedElement	constrainedElement

2.17.3 Constraint Constraints

None.

2.17.4 Constraint Limitations

A MOF Constraint's expression has any type, but a UML Constraint's expression body has string type. Therefore, the profile can support only an expression rendered as a string.

2.18 Generalizes

A UML Generalization maps to a MOF Generalizes link.

2.18.1 Tags on UML Generalization

None.

2.18.2 Generalization-to-Generalizes Property Map

None. Generalizes is an association, not a class.

2.18.3 Generalization-to-Generalizes Constraints

Each UML Generalization within a Model mapped to a MOF Package must connect GeneralizableElements that are also mapped to MOF elements.

2.18.4 Generalization-to-Generalizes Limitations

None.

2.19 Tag

A UML TaggedValue maps to a MOF Tag, except that any UML TaggedValue whose tag is used by this profile is not preserved as a MOF Tag.

2.19.1 Tags on UML TaggedValue

None. A UML TaggedValue cannot be tagged.

2.19.2 TaggedValue-to-Tag Property Map

MOF Property	UML Property or Value
name	modelElement.name* + "." + tag
annotation	""
container	If modelElement is a Model then that Model, otherwise the Model that most immediately owns modelElement
constraints	none
tagId	tag
values	value
elements	modelElement

* if the modelElement is not a Model, then the name is qualified up to but not including the Model that most immediately owns the modelElement — each name is separated by a period (".") character.

2.19.3 TaggedValue-to-Tag Constraints

None.

2.19.4 TaggedValue-to-Tag Limitations

MOF allows a Tag to be contained by an object other than the one it tags. UML requires a tag to be contained by the object it tags. Therefore, when a MOF Tag is rendered in UML, the profile does not retain the relationship to the Tag's container.

UML does not give a name to a TaggedValue other than its tag. Therefore, a MOF Tag name is not preserved when rendered in UML using the profile.

MOF supports any type of value for a Tag. UML supports only a string value. Therefore, the profile supports only string values.

MOF supports having multiple values with a single tag. UML supports only one. Therefore, the profile supports only a single value.

A single MOF tag can be attached to multiple model elements. A UML TaggedValue can be attached to only one. Therefore, the profile supports only a single ModelElement attached to a tag.

The profile does not support annotations, constraints, or tags on tags.

Section IV - Guidelines

This section gives guidelines for designing metamodels using the UML profile for MOF. These guidelines are drawn from several experiences of using UML to design and extend metamodels deployed using MOF.

Refer to the MOF Specification for a comprehensive explanation of MOF.

2.20 Modularity

Separate different modeling areas into different metamodels. Minimize dependencies between metamodels. Make no circular dependencies between them — otherwise valid CORBA IDL interfaces cannot be generated.

The outermost package of a deployed metamodel can be thought of as a type. It is the type of each MOF package extent defined by the metamodel. Avoid nesting metamodels as owned elements so that the metamodels can be deployed in various combinations rather than only as one enormous metamodel. A metamodel can import rather than own other metamodels. Importing gives the same organizational advantage as nesting without imposing strong composition. Metamodels can be imported in two ways: clustered and unclustered. If clustered, an imported metamodel is fully deployed within an extent of the importing metamodel, just as if the imported metamodel had been nested.

Use package inheritance to achieve polymorphism of package extents. If a MOF package inherits from a base package, then an extent of the package can be used wherever an extent of the base package can be used.

2.21 Associations

Give meaningful names to associations, even if you do not display the names in diagrams. The association name is used to define interfaces to access and manage links.

Generally, put an association in the same package as one of its connected classes. If the connected classes are in separate packages, put the association in the most specific package.

When extending an existing model from the outside, feel free to make associations to classes in the existing model. But use existing associations wherever they are appropriate. If you want to draw an association between specific classes for the purpose of showing an existing association between superclasses, then stereotype the association as <<implicit>> so that it is ignored in the mapping to MOF.

MOF does not support association classes or associations having more than two connections. In any case where you would use such an association, model the conceptual association as a class using separate associations for each connection.

2.22 References

A MOF reference is like a derived attribute whose derivation is tied to an association. The values of a reference for an object are the objects linked to that object. Modifying reference values causes links to be added and/or deleted.

When designing a metamodel that extends another from the outside, define references only in the extending metamodel, not in the metamodel it extends.

The definition of a MOF reference affects how package extents contain links. In general, an association link and the objects it connects can all belong to different package extents. However, the MOF Specification defines the Reference Closure Rule which requires any link tied to a reference to be contained by the same package extent as the object having the reference. If an association has references on both ends, both linked objects and the link must all be contained in the same package extent. Before defining a reference, give thought to the Reference Closure Rule so that you do not mistakenly prevent links from interrelating objects across different package extents. Conversely, use a reference where you want to force links to be in the same package extent as the linked objects.

Here is an example. Suppose a metamodel has a class called GE and an association from GE to GE called Generalizes. One end is called supertype and the other is called subtype. Both ends are navigable. If a reference is on both ends, then a link can only connect GE objects within the same package extent. If a reference is only on the subtype end (referring to supertype), then a link must be in the same package extent as its subtype, but it can link to a supertype in the same or a different package extent.

2.23 DataTypes

Avoid defining a complex data type where a class can be used.

Avoid defining enumerations because they limit extensibility. There is no way to extend an enumeration type from an outside metamodel.

2.24 Names

Form multiword names by concatenating words with no intervening spaces, hyphens or underscores. For names of packages, classifiers, and associations upcase the first letter of each word. For names of features and association ends upcase the first letter of each word except for the first word in the name. Do not prefix all of the names in a package with the package name — the package name is already part of the fully qualified name.

Using spaces, punctuation, or leading numerals in names can cause problems for middleware, programming language, and XML bindings.

A

- Application Objects vi
- Association 2-15
- Association Constraints 2-15
- Association Limitations 2-15
- Association Property Map 2-15
- AssociationEnd 2-15
- AssociationEnd Constraints 2-16
- AssociationEnd Limitations 2-16
- AssociationEnd Property Map 2-16
- Associations 2-21
- Attribute 2-9
- Attribute Constraints 2-10
- Attribute Limitations 2-10
- Attribute Property Map 2-10
- Attribute-to-Parameter Constraints 2-14
- Attribute-to-Parameter Limitations 2-14
- Attribute-to-Parameter Property Map 2-14

C

- Class 2-8
- Class Constraints 2-9
- Class Limitations 2-9
- Class Property Map 2-9
- Common Facilities 1-vi
- Compliance Points 1-1
- Conformance Issues 1-1
- Constant 2-18
- Constraint 2-18
- Constraint Constraints 2-19
- Constraint Limitations 2-19
- Constraint Property Map 2-19

D

- DataType 2-16
- DataType Constraints 2-17
- DataType Limitations 2-17
- DataType Property Map 2-17
- DataTypes 2-22
- DataValue-to-Constant Constraints 2-18
- DataValue-to-Constant Limitations 2-18
- DataValue-to-Constant Property Map 2-18

E

- ElementImport-to-Import Constraints 2-8
- ElementImport-to-Import Limitations 2-8
- ElementImport-to-Import Property Map 2-8
- Exception 2-13
- Exception Constraints 2-14
- Exception Limitations 2-14
- Exception Parameter 2-14
- Exception Property Map 2-14
- Explicit Reference Property Map 2-11

G

- Generalization 2-19
- Generalization-to-Generalizes Constraints 2-19
- Generalization-to-Generalizes Limitations 2-19
- Generalization-to-Generalizes Property Map 2-19
- Guidelines 2-21

I

- Implicit Reference Property Map 2-11
- Import 2-8

M

- Metaobject Facility (MOF) 2-4
- ModelElement 2-6
- ModelElement Constraints 2-6
- ModelElement Limitations 2-6
- ModelElement Property Map 2-6
- Model-to-Package Constraints 2-7
- Model-to-Package Limitations 2-8
- Model-to-Package Property Map 2-7
- Modularity 2-21

N

- Names 2-22

O

- Object Management Architecture
 - Reference model for v
- Object Management Group v
- Object Request Broker vi
- Object Services vi
- OMG Modeling vi
- Operation 2-12
- Operation Constraints 2-12
- Operation Limitations 2-12
- Operation Property Map 2-12

P

- Package 2-7
- Parameter 2-13
- Parameter Constraints 2-13
- Parameter Limitations 2-13
- Parameter Property Map 2-13

R

- Reference 2-10
- Reference Constraints 2-11
- Reference Limitations 2-12
- Reference model v
- References 2-22

T

- Tag 2-19
- TaggedValue-to-Tag Constraints 2-20
- TaggedValue-to-Tag Limitations 2-20
- TaggedValue-to-Tag Property Map 2-20
- Tags on Attribute of UML Exception 2-14
- Tags on UML Association 2-15
- Tags on UML AssociationEnd 2-15
- Tags on UML Attribute with No Stereotype 2-9
- Tags on UML Attribute with Stereotype > 2-11
- Tags on UML Class 2-9
- Tags on UML Constraint 2-18
- Tags on UML DataType 2-17
- Tags on UML DataValue 2-18
- Tags on UML ElementImport 2-8
- Tags on UML Exception 2-13
- Tags on UML Generalization 2-19
- Tags on UML ModelElement 2-6

Index

Tags on UML Operation 2-12
Tags on UML Parameter 2-13
Tags on UML TaggedValue 2-20

U

UML Profile for MOF 1-1
Unified Modeling Language (UML) 2-4