XMI Version 1Production of XML Schema Specification

During the finalization phase, the XML Production of XML Schema Final Adopted Specification (ad/2001-12-03) was split into two documents to resolve one of the issues. The documents are:
1) This doument, which defines XML 1 schema production
2) XML Metadata Interchange (XMI) Version 2, which defines XMI 2 document and schema production (ptc/2002-06-03)

The FTF recommends that this document be folded into the XMI Version 1 specification by the XMI 1.3 RTF, and that XMI Version 2 be a stand-alone specification.

The final report for this FTF is available for review (`ptc/2002-06-01`).

XMI Version 1 Production of XML Schema

OMG Document ptc/2002-06-02
June 10, 2002

This sprecification recommended for inclusion in XMI 1.3, currently in revision.

Copyright 1998, 1999, 2000, 2001 IBM Corporation

Copyright 1998, 1999, 2000, 2001 Unisys Corporation

Copyright 1998, 1999, 2000, 2001 Softeam

The companies listed above hereby grant a royalty-free license to the Object Management Group, Inc. (OMG) for worldwide distribution of this document or any derivative works thereof, so long as the OMG reproduces the copyright notices and the below paragraphs on all distributed copies.

The material in this document is submitted to the OMG for evaluation. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE,THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR APARTICULAR PURPOSE. The companies listed above shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. The information contained in this document is subject to change without notice.

This document contains information which is protected by copyright. All Rights Reserved. Except as otherwise provided herein, no part of this work may be reproduced or used in any form or by any meansógraphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systemsó without the permission of one of the copyright owners. All copies of this document must include the copyright and other information contained on this page.

The copyright owners grant member companies of the OMG permission to make a limited number of copies of this document (up to fifty copies) for their internal use as part of the OMG evaluation process.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Right in Technical Data and Computer Software Clause at DFARS 252.227.7013.

CORBA, OMG, and Object Request Broker are trademarks of Object Management Group.

Table of Contents

Preface 1

1.1 Introduction

XMI is a widely used interchange format for sharing objects using XML. Sharing objects in XML is a comprehensive solution that build on sharing data with XML. XMI is applicable to a wide variety of objects: analysis (UML), software (Java, C++), components (EJB, IDL, Corba Component Model), and databases (CWM). Over 30 companies have XMI implementations.

XMI defines many of the important aspects involved in describing objects in XML:

- The representation of objects in terms of XML elements and attributes is the foundation.
- Since objects are typically interconnected, XMI includes standard mechanisms to link objects within the same file or across files.
- Object identity allows objects to be referenced from other objects in terms of IDs and UUIDs.
- The versioning of objects and their definitions is handled by the XMI model.
- Validation of XMI documents using DTDs and Schemas

XMI describes solutions to the above issues by specifying EBNF production rules to create XML documents, DTDs, and Schemas that share objects consistently. XMI Version 1 defines production two kinds of production rules for sharing objects with XML:

- Production of XML DTDs starting from an object model.
- Production of XML documents starting from objects.

With the recent work by the W3C in XML Schemas, a more comprehensive form of XML document validator, this submission adds these production rules:

- Production of XML Schemas starting from an object model: Chapter 2.

MOF is the foundation technology for describing object models, which cover the wide range of object domains: analysis (UML), software (Java, C++), components (EJB, IDL, Corba Component Model), and databases (CWM).

XMI is applicable to all levels of objects and metaobjects. Although this document focuses on MOF metaobjects, general objects may be serialized and interchanged with XMI.

The term "XML document" in this specification is equivalent to a general stream of XML data.

Note – XML Schema is now a full recommendation of the W3C.

1.2 Guide to the Specification

This specification is presented in the following chapters:

Chapter 1 Preface

Introduces the submission and provides the context for the XMI technology within the OMG architecture

Chapter 2 Proof of Concept

Describes proof of concept efforts and results, in demonstration of the proposal’s technical viability.

Chapter 3 Response to RFP Requirements

Identifies the specific RFP requirements and this proposal’s response to each requirement.

Chapter 4 XMI production of XML Schema Design Principles

Provides a discussion of the design of the production of XML Schemas.

Chapter 5 XML Schema Production

Specifies the production rules for XML Schemas in the XMI 1.1 and XMI2 formats.

Chapter 6 XML Document Production

Specifies the production rules for encoding any model in the XMI 2 format.

Chapter 7 Production of MOF from XML

Specifies the production rules to reverse engineer MOF models from XML.

Chapter 8 XML Schema Metamodel

Defines a metamodel for the XML Schema specification.

Chapter 9 Conformance Issues

Discusses conformance - mandatory and optional; compliance points in the XMI specification.

References

Lists the references used in this specification

1.3 Conventions

XML appears using this font.

Caution – Cautionary information appears with this prefix, framing, and in this font.

Note – Items of note appear with this prefix, framing, and in this font

Please note that any change bars have no semantic meaning. They show the places that errata were discovered since the last submission. They are present for the convenience of readers and submitters so that the final edits can be identified.

1.4 Changes Since Initial Submission

The following changes have been made since the initial submission:

1. The schema grammars have been updated to conform to the XML Schema Proposed Recommendation, 16 March 2001.

2. The XMI serialization has been streamlined by reducing the number of XML elements that are required. Previously, there was an XML element to represent a feature and another XML element to represent the feature's value; now there is only one XML element. The XML Document Production Grammar and the XMI 2.0 Schema Production Grammar reflect this change.

3. You may specify tag value pairs in MOF metamodels to tailor the schemas that are produced; you are not required to do so, however.

4. Several of the XMI elements defined by XMI and specified in the header of XMI documents are now defined by a MOF model, so the XML Document Production Grammar does not include special rules to serialize them.

5. Examples of XMI 2 have been included in the text.

6. Reverse engineering for XML Schemas has been added. Reverse engineering of DTDs from XML has been removed.

1.5 Changes Since Revised Submission

1. XML schema is now a full recommendation of the W3C. Small changes have been made to match the final syntax.

2. XML Schema Model has been added.

3. The XMI model has been changed to more succinctly encode differences.

XML Schema Production 2

2.1 Purpose

This section describes the rules for creating an XMI Version 1 schema from a MOF-based metamodel. You can use either an XMI Version 1 DTD or an XMI Version 1 schema to validate any XMI Version 1 document. The conformance rules are stated in Chapter 3.

Notation for EBNF

The rule sets are stated in EBNF notation. Each rule is numbered for reference. Rules are written as rule number, rule name, for example 1a. SchemaStart. Text within quotation marks are literal values, for example "<xsd:element". Text enclosed in double slashes represents a placeholder to be filled in with the appropriate external value, for example //Name of Attribute//. Literals should be enclosed in single or double quotation marks when used as the values for XML attributes in XML documents. The suffix "*" is used to indicate repetition of an item 0 or more times. The suffix "?" is used to indicate repetition of an item 0 or 1 times. The suffix "+" is used to indicate repetition of an item 1 or more times. The vertical bar "|" indicates a choice between two items. Parentheses "()" are used for grouping items together.

EBNF ignores white space; hence these rules do not specify white space treatment. However, since white space in XML is significant, the actual schema generation process must insert white space at the appropriate points.

The XML element names generated for XMI Version 1 schemas are *qualified* names. A qualified name consists of a Class, Package, or Association name. Attributes or References are further prefixed by a period (".") delimiter. You may use XML namespaces with the qualified names.

2.2 XMI Version 1 Schemas

2.2.1 EBNF

The EBNF for XMI Version 1 schemas is listed below with rule descriptions between sections:

```
1.  Schema              ::= 1a:SchemaStart
                            1d:ImportsAndIncludes?
                            1e:FixedDeclarations
                            2:PackageSchema+
                            1f:SchemaEnd
1a. SchemaStart         ::= "<xsd:schema
                                xmlns:xsd='http://www.w3.org/2001/XMLSchema'"
                               <1b:NamespaceDecl>*
                               <1c:TargetNamespace>?
                              ">"
1b. NamespaceDecl       ::= "xmlns:" //Namespace name// "="
                              "'" //Namespace URI// "'"
1c. TargetNamespace     ::= "targetNamespace='" //Namespace URI// "'"
1d. ImportsAndIncludes ::= // Imports and includes //
1e. FixedDeclarations   ::= //Fixed declarations//
1f. SchemaEnd           ::= "</xsd:schema>"
1g. XMIFixedAttribs     ::= "<xsd:attributeGroup ref='XMIIdentityAttribs'/>
                              <xsd:attributeGroup ref='XMILinkAttribs'/>"
1h. Namespace           ::= ( //Name of namespace// ":" )?
```

1. A schema consists of a schema XML element that contains import and include statements, fixed declarations, plus declarations for the contents of the Packages in the metamodel.

1a. The schema XML element consists of the schema namespace attribute, namespace attributes for the other namespaces used in the schema, if any, and an optional target namespace attribute. These rules are written as if the namespace name for the schema namespace is "xsd", but you can substitute another name for the schema namespace name and still conform to this specification.

1b. Each namespace used in the schema must have a namespace attribute that identifies the namespace name and the namespace URI. If the namespace name is "" the attribute name should be "xmlns".

1c. If the schema has a target namespace, the targetNamespace attribute is present.

1d. If the schema uses declarations from other schemas, the appropriate include or import statements must be present.

1e. The fixed declarations are listed in section 5.2.2. They can either be physically present in the schema or imported or included into the schema.

1f. The end of the schema XML element.

1g. The fixed XMI attributes present on the major elements provide element identity and element linking.

1h. A namespace is a namespace name followed by a ":". If no namespace name is given, the rule is a blank.

```
2.  PackageSchema    ::= ( 2:PackageSchema
                              | 3:ClassSchema
                              | 4:AttributeElmtDef
                              | 7:CompositionSchema
                              | 10:AssociationSchema
                              | 13:EnumSchema)*
                            9:PackageElementDef
```

2. The schema contribution from a Package consists of the declarations for any contained Packages, Classes, classifier level Attributes, containment aggregations, Associations without References, enumerations, and an XML element declaration for the Package itself.

```
3.  ClassSchema      ::= ( 4:AttributeElmtDef | 5:ReferenceElmtDef )*
                             6:ClassElementDef
```

3. The class schema contribution consists of the element definitions for any Attributes and References of the Class and an element definition for the Class itself.

```
4.  AttributeElmtDef ::= "<xsd:element name='" 4a:AttribElmtName "'>"
                            4c:AttribContents
                            "</xsd:element>"
4a. AttribElmtName   ::= 6a:ClassElmtName "." 4b:AttribName
4b. AttribName       ::= //Name of Attribute//
4c. AttribContents   ::= 4d:AttribData
                            | 4e:AttribEnum
                            | 4f:AttribClasses
4d. AttribData       ::= "<xsd:complexType mixed='true'>
                             <xsd:sequence>
                               <xsd:element ref='XMI.reference' minOccurs='0'
                                              maxOccurs='unbounded'/>
                             </xsd:sequence>
                            </xsd:complexType>"
4e. AttribEnum       ::= "<xsd:complexType>
                             <xsd:attribute name='xmi.value'
                                             type='" 13a:EnumTypeName "'"
                                            "use='required'>
                            </xsd:complexType>"
4f. AttribClasses    ::= "<xsd:complexType>
                            <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                            4g:AttribClass
                            "</xsd:choice>
                             </xsd:complexType>"
4g. AttribClass     ::= ("<xsd:element ref='" 6a:ClassElmtName "'/>")+
```

4. These rules define the declaration of an Attribute of a Class in the metamodel as an XML element. These metamodel Attributes can, in some cases, be expressed as XML attributes rather than XML elements. This is specified in rule 6h and gives the document writer the ability to choose which representation is most convenient in a particular use in an XML document.

4a, 4b. The name of the XML element representing an Attribute of a Class is the element name of the Class containing the Attribute followed by a dot separator and the name of the Attribute.

4c. An Attribute which can be expressed as a data value is expressed in terms of a string or reference to its content (4d) or an enumeration (4e, 4g, 4h). An Attribute which has a Class as its value is expressed in terms of the possible Class types that can be instances of its value (4f). If the Class has subclasses, the element name of each of its subclasses is included in the declaration.

Note – If the MOF Tag "org.omg.xmi.enumerationUnprefix" is attached to the DataType where the enumerated values of the Attribute are defined, the value of this Tag contains a prefix which will be removed from the values of enumeration literals before they are written in the schema.

Note – Although the schema as produced by this grammar cannot restrict the interspersing of other Attribute values among the instances of the values of a multi-valued Attribute, the XML document production rules state that all values for the Attribute should be consecutive elements and not interspersed with other Attribute values.

```
5.  ReferenceElmtDef  ::= "<xsd:element name='" //Name of class// "."
                                               5b:ReferenceName "'>"
                           "<xsd:complexType>
                          <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                             5c:RefContents
                           "</xsd:choice>
                            </xsd:complexType>
                            </xsd:element>"
5a. ReferenceElmtName ::= 6a:ClassElmtName "." 5b:ReferenceName
5b. ReferenceName     ::= //Name of Reference//
5c. RefContents       ::= ("<xsd:element ref='" 6a:ClassElmtName "'/>")+
```

5. These rules define the declaration of a metamodel Reference as XML element content for linking by proxy. It is also possible to place the Reference in an XML attribute, as defined in rule 6i. This provides the ability to more conveniently represent References when the limited linking facilities available in such a case are sufficient.

5a, 5b. The name of the XML element representing a Reference is the element name of the Class containing the Reference, a dot separator, and the name of the Reference.

5c. The element name of the type of the Reference is given in the declaration. Any subclass of the type can, but need not, appear in the declaration as well. An XML linkage to a Class element will work if the target of the linkage is a member of a Class or one of its subclasses.

```
6.  ClassElementDef   ::= "<xsd:element name='" //Name of class// "'>"
                            "<xsd:complexType>
                           <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                            6b:ClassContents
                             "</xsd:choice>"
                            6g:ClassAttListItems
                            "</xsd:complexType>
                             </xsd:element>"
6a. ClassElmtName     ::= 1h:Namespace //Name of Class//
6b. ClassContents     ::= 6d:ClassAttributes
                          6e:ClassReferences
                          6f:ClassCompositions
                          6c:Extension
6c. Extension         ::= "<xsd:element ref='XMI.extension'/>"
6d. ClassAttributes   ::= ("<xsd:element ref='" 4a:AttribElmtName "'/>")+
6e. ClassReferences   ::= ( "<xsd:element ref='" 5a:ReferenceElmtName
                              "'/>" )+
6f. ClassCompositions ::= ( "<xsd:element ref='" 6a:ClassElmtName
                               "'/>" )+
6g. ClassAttListItems ::= 1g:XMIFixedAttribs 6h:ClassAttribAtts
6h. ClassAttribAtts   ::= ( 6i:ClassAttribRef
                            | 6j:ClassAttribData
                            | 6k:ClassAttribEnum )*
6i. ClassAttribRef    ::= "<xsd:attribute name='" 4b:AttribName "'"
                                "type='xsd:IDREFS' use='optional'/>"
6j. ClassAttribData   ::= "<xsd:attribute name='" 4b:AttribName "'"
                                 "type='xsd:string' "
                                 "use='optional'
                                 (default='" 6l:ClassAttribDflt "'")?
                                "/>"
6k. ClassAttribEnum   ::= "<xsd:attribute name='" 4b:AttribName "'"
                                "type='" 13a:EnumTypeName "'"
                                "use='optional'
                                (default='" 6l:ClassAttribDflt "'") ?
                                "/>"
6l. ClassAttribDflt   ::= //Default value//
```

6. These rules describe the declaration of a Class in the metamodel as an XML element with XML attributes.

6a. The name of references to the XML element for the Class is the name of the Class prefixed by the namespace, if present.

6b, 6c. The XML element for the Class contains XML elements for the contained non-derived Attributes, References and Compositions of the Class, plus an extension element.

6d. The XML element name for each non-derived Attribute of the Class is listed as part of the content model of the Class element. This includes the Attributes defined for the Class itself as well as all of the non-derived Attributes inherited from superclasses of the Class.

6e. The XML element name for each Reference of the Class is listed in the content model of the Class. The list includes the References defined for the Class itself, as well as all References inherited from the superclasses of the Class.

6f. The XML element name for each Class contained in this Class in the content model of the Class element. Here, containment means that the contained Class is directly owned as an ownedElement of the Class. In addition to the element name of the contained Class, the element name of each subclass of the contained Class must also be listed.

6g, 6h. In addition to the standard identification and linkage attributes, the attribute list of the Class element can contain XML attributes for the Attributes and non-composite References of the Class, when the limited facilities of the XML attribute syntax allow expression of the necessary values.

6i. References (either directly owned by the Class or inherited) can be expressed as XML id reference XML attributes.

6j. Single-valued Attributes (direct or inherited) of a Class that have a string representation for their data are mapped to CDATA XML attributes. Multi-valued Attributes of a Class cannot be so expressed, since the XML attribute syntax does not allow repetition of values.

6k. Single-valued Attributes (direct or inherited) that have enumerated values are mapped to enumerated XML attributes in the same manner as in an AttributeElmtDef (4, 4e).

6l. If an Attribute is expressed as an XML attribute, its default value may be expressed in the schema if there is a MOF Tag "org.omg.xmi.defaultValue" attached to the Attribute. The value of this Tag must be expressible as an XML attribute string.

```
7.  CompositionSchema    ::= 8:CompositionElmtDef*
```

7. Elements for Associations that represent compositions are described using rule 8.

```
8.  CompositionElmtDef ::= "<xsd:element name='" //Name of class//
                                           "." 5b:ReferenceName "'>"
                            "<xsd:complexType>
                           <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                           6f:ClassCompositions
                             "</xsd:choice>
                              </xsd:complexType>
                              </xsd:element>"
```

8. The composition element is generated for each Reference in the Package which has an exposedEnd whose aggregation is composite. This element is used in the class contents XML element (6). It is a list of the Class which is the type of the Reference, as well as all of its subclasses. The name of the Reference XML element is the element name of the Class containing the Reference, followed by a dot and the name of the Reference.

```
9.  PackageElementDef ::= "<xsd:element name='" 9b:PkgName "'>"
                            "<xsd:complexType>
                           <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                            9c:PkgContents
                            "</xsd:choice>"
                           9h:PkgAttListItems
                            "</xsd:complexType>
                             </xsd:element>"
9a. PkgElmtName       ::= 1h:Namespace 9b:PkgName
9b. PkgName           ::= //Name of Package//
9c. PkgContents       ::= 9d:PkgAttributes ?
                          9e:PkgClasses ?
                          9f:PkgAssociations ?
                          9g:PkgPackages ?
                          6c:Extension
9d. PkgAttributes     ::= ( "<xsd:element ref='" 4a:AttribElmtName "'/>")+
9e. PkgClasses        ::= ( "<xsd:element ref='" 6a:ClassElmtName "'/>")+
9f. PkgAssociations   ::= ( "<xsd:element ref='" 12a:AssnElmtName "'/>")+
9g. PkgPackages       ::= ( "<xsd:element ref='" 9b:PkgElmtName "'/>")+
9h. PkgAttListItems   ::= 1g:XMIFixedAttribs 9i:PkgAttribAtts
9i. PkgAttribAtts     ::= 6h:ClassAttribAtts
```

9. The schema contribution from the Package consists of an XML element definition for the Package, with a content model specifying the contents of the Package.

9a, 9b. The name of the Package XML element.

9c. The Package contents consists of any classifier level Attributes, Associations without References, Classes, nested Packages and an extension.

9d. Classifier level Attributes of a Package are also known as static attributes. Such Attributes inherited from Packages from which this Package is derived are also included.

9e. Each Class in the Package is listed. Classes contained in Packages from which this Package is derived are also included.

9f. It is possible that the Package contains Associations which have no References, i.e. no Class contains a Reference which refers to an AssociationEnd owned by the Association. Every such Association contained in the Package or Package from which the Package is derived is listed as part of the Package contents in order that its information can be transmitted as part of the XML document.

9g. Nested Packages are listed. Nested Packages included in Packages from which this Package is derived are also included.

9h, 9i. The XML attributes for the Package are the fixed identity and linking XML attributes.

```
10. AssociationSchema    ::= 11:AssociationEndDef
                             11:AssociationEndDef
                             12:AssociationDef
```

10. The XML elements for unreferenced Associations consist of definitions for its AssociationEnds and for the Association itself.

```
11. AssociationEndDef ::= "<xsd:element name='" 12b:AssnName "."
                                                  11b:AssocEndName "'>
                             <xsd:complexType>"
                            11c:AssocEndAtts
                             "</xsd:complexType>
                              </xsd:element>"
11a. AssocEndElmtName  ::= 12b:AssnElmtName "." 11b:AssocEndName
11b. AssocEndName      ::= //Name of AssociationEnd//
11c. AssocEndAtts      ::= 1g:XMIFixedAttribs
```

11. The declaration for an AssociationEnd XML element has no content model, though it has the standard set of XML attributes.

11a, 11b. The name of the AssociationEnd XML element is the element name of the association containing the AssociationEnd, a dot separator, and the name of the AssociationEnd.

11c. The fixed identity and linking XML attributes are the AssociationEnd's only XML attributes.

```
12. AssociationDef    ::= "<xsd:element name='" 12b:AssnName "'>"
                            "<xsd:complexType>
                           <xsd:choice minOccurs='0' maxOccurs='unbounded'>"
                           12c:AssnContents
                             "</xsd:choice>"
                           12d:AssnAtts
                               "</xsd:complexType>
                                 </xsd:element>"
12a. AssnElmtName       ::= 1h:Namespace 12b:AssnName
12b. AssnName           ::= //Name of Association//
12c. AssnContents       ::= "<xsd:element ref='" 11a:AssocEndElmtName "'/>"
                              "<xsd:element ref='" 11a:AssocEndElmtName "'/>"
                            6c:Extension
12d. AssnAtts           ::= 1g:XMIFixedAttribs
```

12, 12c. The declaration of an unreferenced Association consists of the names of its AssociationEnd XML elements.

12a, 12b. The name of the XML element representing the Association.

12d. The fixed identity and linking XML attributes are the Association XML attributes.

```
13.  EnumSchema       ::= "<xsd:simpleType name='" 13b:EnumName "'>"
                               "<xsd:restriction base='xsd:string'>"
                                  13c:EnumLiterals
                               "</xsd:restriction>"
                             "</xsd:simpleType>"
13a. EnumTypeName     ::= 1h:Namespace 13b:EnumName
13b. EnumName         ::= // Name of enumeration //
13c. EnumLiterals      ::= ("<xsd:enumeration value='" 13d:EnumLiteral "'/>")+
13d. EnumLiteral      ::= // Name of enumeration literal //
```

13. The enumeration schema contribution consists of a simple type derived from string whose legal values are the enumeration literals.

13a. The name of the enumeration type in schema references

13c. Each enumeration literal is put in the value XML attribute of an enumeration XML element.

13d. The name of the enumeration literal

2.2.2 Fixed Schema Declarations

There are some elements of the schema which are fixed, constituting a form of "boilerplate" necessary for every MOF schema. These elements are described in this section. They should be physically included at the beginning of the generated schema, or placed in their own schema and included or imported into the generated schema.

Only the schema content of the fixed declarations is given here. For a complete description of the semantics of these declarations, see the XMI Version 1 specification.

The fixed declarations are included in three separate xsd files, since the linking attributes use the "xlink" and "xml" namespaces.

Here is the xmi11.xsd schema, which imports the xlink.xsd and xml.xsd schemas:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns='http://www.w3.org/2001/XMLSchema'
         xmlns:xlink="http://www.w3.org/XLink"
         xmlns:xml="http://www.w3.org/XML">

<annotation>
   <documentation>
      This schema, used in conjunction with XMI Version 1 schemas
      will validate the same set of documents as XMI Version 1 DTDs.
   </documentation>
</annotation>

<import namespace="http://www.w3.org/XLink" schemaLocation="xlink.xsd"/>
<import namespace="http://www.w3.org/XML" schemaLocation="xml.xsd"/>

<element name="XMI">
   <annotation>
      <documentation>
         This element declaration should match the DTD declaration:
            !ELEMENT XMI (XMI.header?, XMI.content?, XMI.difference*,
                              XMI.extensions*)
            !ATTLIST XMI
                            xmi.version CDATA #FIXED "1.1"
                            timestamp CDATA #IMPLIED
                            verified (true | false) #IMPLIED
      </documentation>
   </annotation>
   <complexType>
```

```
        <sequence>
            <element name="XMI.header" type="header" minOccurs="0"
maxOccurs="1"/>
            <element name="XMI.content" minOccurs="0" maxOccurs="1">
                <complexType>
                    <sequence>
                        <any minOccurs="0" maxOccurs="unbounded"/>
                    </sequence>
                </complexType>
            </element>
            <element name="XMI.difference" type="difference" minOccurs="0"
maxOccurs="unbounded"/>
            <element name="XMI.extensions" type="extensions" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="xmi.version" type="string" use="required"
fixed="1.1"/>
        <attribute name="timestamp" type="string" use="optional"/>
        <attribute name="verified" type="boolean" use="optional"/>
    </complexType>
</element>

<complexType name="header">
    <annotation>
        <documentation>
            This element declaration should match the DTD declaration:
                !ELEMENT XMI.header (XMI.documentation?, XMI.model*,
XMI.metamodel*,
                                        XMI.metametamodel*, XMI.import*)
        </documentation>
    </annotation>
    <sequence>
        <element name="XMI.documentation" type="documentation"
minOccurs="0" maxOccurs="1"/>
        <element name="XMI.model" type="model" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="XMI.metamodel" type="model" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="XMI.metametamodel" type="model" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="XMI.import" type="model" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="documentation" mixed="true">
    <annotation>
        <documentation>
```

```
          This type definition should match the DTD declarations:
             !ELEMENT XMI.documentation (#PCDATA | XMI.owner |
XMI.contact |
                                              XMI.longDescription |
XMI.shortDescription |
                                              XMI.exporter |
XMI.exporterVersion |
                                              XMI.notice)*
             !ELEMENT XMI.owner ANY
             !ELEMENT XMI.contact ANY
             !ELEMENT XMI.longDescription ANY
             !ELEMENT XMI.shortDescription ANY
             !ELEMENT XMI.exporter ANY
             !ELEMENT XMI.exporterVersion ANY
             !ELEMENT XMI.notice ANY
        </documentation>
    </annotation>
    <choice minOccurs="0" maxOccurs="unbounded">
        <element name="XMI.owner">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.contact">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.longDescription">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.shortDescription">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.exporter">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.exporterVersion">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <element name="XMI.notice">
            <complexType mixed="true">
                <sequence>
                    <any minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
    </choice>
</complexType>

<complexType name="model" mixed="true">
    <annotation>
        <documentation>
            This type definition is used for elements XMI.model,
            XMI.metamodel, XMI.metametamodel, and XMI.import.  It
            should be equivalent to an element with content model of
            ANY and the following attributes:
                %XMI.link.att;
                xmi.name      CDATA #REQUIRED
                xmi.version   CDATA #IMPLIED
        </documentation>
    </annotation>
    <sequence>
        <any minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attributeGroup ref="XMILinkAttribs"/>
    <attribute name="xmi.name" type="string" use="required"/>
    <attribute name="xmi.version" type="string" use="optional"/>
</complexType>
```

```
<complexType name="extensions" mixed="true">
    <annotation>
        <documentation>
            This type definition corresponds to the following declaration:
                !ELEMENT XMI.extensions ANY
                   !ATTLIST XMI.extensions
                          xmi.extender CDATA #REQUIRED
        </documentation>
    </annotation>
    <sequence>
        <any minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="xmi.extender" type="string" use="required"/>
</complexType>

<element name="XMI.extension" type="extension"/>

<complexType name="extension" mixed="true">
    <annotation>
        <documentation>
              This type definition corresponds to the content model and
              attributes of the following element declaration:
                !ELEMENT XMI.extension ANY
                !ATTLIST XMI.extension
                                  %XMI.element.att;
                                  %XMI.link.att;
                                  xmi.extender CDATA #REQUIRED
                                  xmi.extenderID CDATA #IMPLIED
        </documentation>
    </annotation>
    <sequence>
        <any minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attributeGroup ref="XMIIdentityAttribs"/>
    <attributeGroup ref="XMILinkAttribs"/>
    <attribute name="xmi.extender" type="string" use="required"/>
    <attribute name="xmi.extenderID" type="string" use="optional"/>
</complexType>

<complexType name="difference">
    <annotation>
        <documentation>
            This type definition corresponds to the content model and
```

```
                attributes of the following element declaration:
               !ELEMENT XMI.difference (XMI.difference | XMI.delete | XMI.add |
                                                    XMI.replace)*
                     !ATTLIST XMI.difference
                                         %XMI.element.att;
                                         %XMI.link.att;
            </documentation>
        </annotation>
        <choice minOccurs="0" maxOccurs="unbounded" >
            <element name="XMI.difference" type="difference"/>
            <element name="XMI.delete" type="delete"/>
            <element name="XMI.add" type="add"/>
            <element name="XMI.replace" type="replace"/>
        </choice>
        <attributeGroup ref="XMIIdentityAttribs"/>
        <attributeGroup ref="XMILinkAttribs"/>
    </complexType>

    <complexType name="delete">
        <annotation>
            <documentation>
                This type definition corresponds to the content model and
                attributes of the following element declaration:
                    !ELEMENT XMI.delete EMPTY
                    !ATTLIST XMI.delete
                                   %XMI.element.att;
                                   %XMI.link.att;
            </documentation>
        </annotation>
        <attributeGroup ref="XMIIdentityAttribs"/>
        <attributeGroup ref="XMILinkAttribs"/>
    </complexType>

    <complexType name="add" mixed="true">
        <annotation>
            <documentation>
                This type definition corresponds to the content model and
                attributes of the following element declaration:
                      !ELEMENT XMI.add ANY
                      !ATTLIST XMI.add
                                  %XMI.element.att;
                                  %XMI.link.att;
                                  xmi.position CDATA "-1"
            </documentation>
        </annotation>
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attributeGroup ref="XMIIdentityAttribs"/>
        <attributeGroup ref="XMILinkAttribs"/>
       <attribute name="xmi.position" type="string" use="optional" default="-
    1"/>
    </complexType>

    <complexType name="replace">
        <annotation>
            <documentation>
                This type definition corresponds to the content model and
                attributes of the following element declaration:
                      !ELEMENT XMI.replace ANY
                      !ATTLIST XMI.replace
                                  %XMI.element.att;
                                  %XMI.link.att;
                                  xmi.position CDATA "-1"
            </documentation>
        </annotation>
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attributeGroup ref="XMIIdentityAttribs"/>
        <attributeGroup ref="XMILinkAttribs"/>
       <attribute name="xmi.position" type="string" use="optional" default="-
    1"/>
    </complexType>

    <attributeGroup name="XMIIdentityAttribs">
        <annotation>
            <documentation>
                This attribute group corresponds to the following entity:
                     !ENTITY % XMI.element.att
                          'xmi.id ID #IMPLIED xmi.label CDATA #IMPLIED xmi.uuid
                             CDATA #IMPLIED '
            </documentation>
        </annotation>
        <attribute name="xmi.id" type="ID" use="optional"/>
        <attribute name="xmi.label" type="string" use="optional"/>
        <attribute name="xmi.uuid" type="string" use="optional"/>
    </attributeGroup>
```

```
<attributeGroup name="XMILinkAttribs">
    <annotation>
        <documentation>
             This attribute group corresponds to the following entity:
               !ENTITY % XMI.link.att
                  'href CDATA #IMPLIED xmi.idref IDREF #IMPLIED xml:link
                   CDATA #IMPLIED xlink:inline (true | false) #IMPLIED
                  xlink:actuate (show | user) #IMPLIED xlink:content-role
                   CDATA #IMPLIED xlink:title CDATA #IMPLIED xlink:show
                   (embed | replace | new) #IMPLIED xlink:behavior CDATA
                   #IMPLIED'
        </documentation>
    </annotation>
    <attribute name="href" type="string" use="optional"/>
    <attribute name="xmi.idref" type="IDREF" use="optional"/>
    <attribute ref="xml:link" use="optional"/>
    <attribute ref="xlink:inline" use="optional"/>
    <attribute ref="xlink:actuate" use="optional"/>
    <attribute ref="xlink:content-role" use="optional"/>
    <attribute ref="xlink:title" use="optional"/>
    <attribute ref="xlink:show" use="optional"/>
    <attribute ref="xlink:behavior" use="optional"/>
</attributeGroup>

<element name="XMI.reference" type="reference"/>

<complexType name="reference" mixed="true">
    <annotation>
        <documentation>
            This type definition corresponds to the content model and
            attributes of the following element declaration:
                  !ELEMENT XMI.reference ANY
                  !ATTLIST XMI.reference
                                        %XMI.link.att;
        </documentation>
    </annotation>
    <sequence>
        <any minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attributeGroup ref="XMILinkAttribs"/>
</complexType>

</schema>
```

```
</complexType>
```

The xlink.xsd schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.w3.org/XLink"
            targetNamespace="http://www.w3.org/XLink">

<xsd:annotation>
  <xsd:documentation>
    The following attribute declarations are for XLink attributes for
    XMI Version 1 schemas.
  </xsd:documentation>
</xsd:annotation>

<xsd:attribute name="inline" type="xsd:boolean"/>

<xsd:attribute name="actuate">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="show"/>
         <xsd:enumeration value="user"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:attribute>

<xsd:attribute name="content-role" type="xsd:string"/>

<xsd:attribute name="title" type="xsd:string"/>

<xsd:attribute name="show">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="embed"/>
         <xsd:enumeration value="replace"/>
         <xsd:enumeration value="new"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:attribute>

<xsd:attribute name="behavior" type="xsd:string"/>

</xsd:schema>
```

The xml.xsd schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.w3.org/XML"
            targetNamespace="http://www.w3.org/XML">

<xsd:annotation>
  <xsd:documentation>
    The following attribute declarations are for XLink attributes for XMI
     1.1 schemas.
  </xsd:documentation>
</xsd:annotation>

<xsd:attribute name="link" type="xsd:string"/>

</xsd:schema>
```

2.2.3 Optional Fixed Schema Declarations

The following fixed schema declarations are used only when required by the metamodel. These are only used for MOF 1.3 models, but not for MOF 1.4 models.

```
<element name="XMI.TypeDefinitions">
    <complexType mixed="true">
      <sequence>
          <any minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
</element>

<element name="XMI.field">
    <complexType mixed="true">
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</element>

<element name="XMI.seqItem">
    <complexType mixed="true">
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</element>
```

```
<element name="XMI.octetStream" type="string"/>

<element name="XMI.unionDiscrim">
    <complexType mixed="true">
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</element>

<element name="XMI.enum">
    <complexType>
      <attribute name="xmi.value" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.any">
    <complexType mixed="true">
        <sequence>
            <any minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attributeGroup ref="XMILinkAttribs"/>
        <attribute name="xmi.type" type="string" use="optional"/>
        <attribute name="xmi.name" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTypeCode">
    <complexType>
        <choice>
            <element ref="XMI.CorbaTcAlias"/>
            <element ref="XMI.CorbaTcStruct"/>
            <element ref="XMI.CorbaTcSequence"/>
            <element ref="XMI.CorbaTcArray"/>
            <element ref="XMI.CorbaTcEnum"/>
            <element ref="XMI.CorbaTcUnion"/>
            <element ref="XMI.CorbaTcExcept"/>
            <element ref="XMI.CorbaTcString"/>
            <element ref="XMI.CorbaTcWstring"/>
            <element ref="XMI.CorbaTcShort"/>
            <element ref="XMI.CorbaTcLong"/>
            <element ref="XMI.CorbaTcUshort"/>
            <element ref="XMI.CorbaTcUlong"/>
            <element ref="XMI.CorbaTcFloat"/>
            <element ref="XMI.CorbaTcDouble"/>
            <element ref="XMI.CorbaTcBoolean"/>
            <element ref="XMI.CorbaTcChar"/>
            <element ref="XMI.CorbaTcWchar"/>
            <element ref="XMI.CorbaTcOctet"/>
            <element ref="XMI.CorbaTcAny"/>
            <element ref="XMI.CorbaTcTypeCode"/>
            <element ref="XMI.CorbaTcPrincipal"/>
            <element ref="XMI.CorbaTcNull"/>
            <element ref="XMI.CorbaTcVoid"/>
            <element ref="XMI.CorbaTcLongLong"/>
            <element ref="XMI.CorbaTcLongDouble"/>
        </choice>
        <attributeGroup ref="XMIIdentityAttribs"/>
    </complexType>
</element>

<element name="XMI.CorbaTcAlias">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTypeCode"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTcStruct">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTcField" minOccurs="0"
                     maxOccurs="unbounded"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTcField">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTypeCode"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcSequence">
    <complexType>
        <choice>
            <element ref="XMI.CorbaTypeCode"/>
            <element ref="XMI.CorbaRecursiveType"/>
        </choice>
        <attribute name="xmi.tcLength" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaRecursiveType">
    <complexType>
        <attribute name="xmi.offset" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcArray">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTypeCode"/>
        </sequence>
        <attribute name="xmi.tcLength" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcObjRef">
    <complexType>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTcEnum">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTcEnumLabel"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
```

```
</element>

<element name="XMI.CorbaTcEnumLabel">
    <complexType>
        <attribute name="xmi.tcName" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcUnionMbr">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTypeCode"/>
            <element ref="XMI.any"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcUnion">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTypeCode"/>
            <element ref="XMI.CorbaTcUnionMbr" minOccurs="0"
                      maxOccurs="unbounded"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTcExcept">
    <complexType>
        <sequence>
            <element ref="XMI.CorbaTcField" minOccurs="0"
                      maxOccurs="unbounded"/>
        </sequence>
        <attribute name="xmi.tcName" type="string" use="required"/>
        <attribute name="xmi.tcId" type="string" use="optional"/>
    </complexType>
</element>

<element name="XMI.CorbaTcString">
    <complexType>
        <attribute name="xmi.tcLength" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcWstring">
    <complexType>
        <attribute name="xmi.tcLength" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcFixed">
    <complexType>
        <attribute name="xmi.tcDigits" type="string" use="required"/>
        <attribute name="xmi.tcScale" type="string" use="required"/>
    </complexType>
</element>

<element name="XMI.CorbaTcShort">
    <complexType/>
</element>

<element name="XMI.CorbaTcLong">
    <complexType/>
</element>

<element name="XMI.CorbaTcUshort">
    <complexType/>
</element>

<element name="XMI.CorbaTcUlong">
    <complexType/>
</element>

<element name="XMI.CorbaTcFloat">
    <complexType/>
</element>

<element name="XMI.CorbaTcDouble">
    <complexType/>
</element>

<element name="XMI.CorbaTcBoolean">
    <complexType/>
</element>
```

```
<element name="XMI.CorbaTcChar">
    <complexType/>
</element>

<element name="XMI.CorbaTcWchar">
    <complexType/>
</element>

<element name="XMI.CorbaTcOctet">
    <complexType/>
</element>

<element name="XMI.CorbaTcAny">
    <complexType/>
</element>

<element name="XMI.CorbaTcTypeCode">
    <complexType/>
</element>

<element name="XMI.CorbaTcPrincipal">
    <complexType/>
</element>

<element name="XMI.CorbaTcNull">
    <complexType/>
</element>

<element name="XMI.CorbaTcVoid">
    <complexType/>
</element>

<element name="XMI.CorbaTcLongLong">
    <complexType/>
</element>

<element name="XMI.CorbaTcLongDouble">
    <complexType/>
</element>
```

Conformance Issues 3

3.1 Introduction

This section describes the required and optional points of compliance with the XMI specification. "XMI Document" and "XMI Schema" are defined as documents and schemas produced by the XMI production (document and XML schema) rules defined in this specification.

3.2 Required Compliance

3.2.1 XMI Schema Compliance

FTF issue 4663: clarify equivalence.

- XMI Schemas must be equivalent to those generated by the XMI Schema production rules specified in this document. Equivalence means that XMI documents that are valid under the XMI Schema production rules would be valid in a conforming XMI Schema and that those XMI documents that are not valid under the XMI Schema production rules are not valid in a conforming XMI Schema.

FTF issue 4706: add section 3.2.2.

3.2.2 Software Compliance

Software is XMI schema compliant when it produces XML schemas that are XMI schema compliant.

3.3 Optional Compliance Points

3.3.1 XMI Extension and Differences Compliance

XMI Documents optionally conform to the following points:

- The guidelines for using the extension elements suggested in the *XML Metadata Interchange (XMI)*, version 2 specification, Section 2.5, *XMI Model* and Section 2.11, *Tailoring Schema Production*. Tools should place their extended information within the designated extension areas, declare the nature of the extension using the standard XMI elements where applicable, and preserve the extensions of other tools where appropriate.
- Processing of XMI differencing elements (See the *XML Metadata Interchange (XMI)*, version 2 specification, Section 2.12, *Transmitting Metadata Differences*) is an optional compliance point.