Specification of Ecore to XML Persistence Mapping

Mark Brörkens[1], Yue Ma[2], Stephan Eberle[2]

[1]itemis AG
[2]itemis France SAS

December 18, 2013

Contents

1 Abstract

In order to enable interoperability between systems and organization it is required to agree on a common metamodel and a common representation that is used for exchange. This document describes generic use cases and best practices for definition of metamodels and XML based data exchange formats. Additionally, it specifies a highly configurable mapping between models and XML documents and XML schema that covers many existing XML mappings such as OMG XMI 1.x and 2.x as well as OMG ReqIF and AUTOSAR.

2 Introduction

Definition of domain specific languages by means of meta models has become common practice in standardization organizations such as AUTOSAR, HIS, EAST-ADL. Each organization has defined its own meta modeling rules and specifications for mapping the meta model to XML Schema. Additionally, it often takes a pretty

long time before tool vendors are able to provide tools that support the new standard.

This document describes a highly configurable framework that allows deriving computer readable XSD schema files from their metamodel as well as an implementation of a tool framework that implements that meta model as well as the XML serialization.

See figures 1 and 2.

Figure 1 depicts the overall work

[To do: add motivation: interoperability]
[To do: add motivation: performance, memory consumption]

3 Use Cases

[To do: overview]

3.1 Variant Handling

[To do: Formulate the use cases as agile user stories]
The XML persistence mapping shall support variant handling on file level. E.g. it shall be possible to store the overall model in several files and to select a variant by exchanging individual XML files.

3.2 Machine Readability

[To do: complex systems as well as simple scripts]

3.3 Human Readability

As an engineer I want to be able to create, read, update and delete XML files according to the XML persistence mapping rules so that I can validate and customize the overall model manually a generic XML editor in case no high-level tools are available yet.
Pattern that support human readability are:

- wrapper XML elements that support folding and unfolding of lists

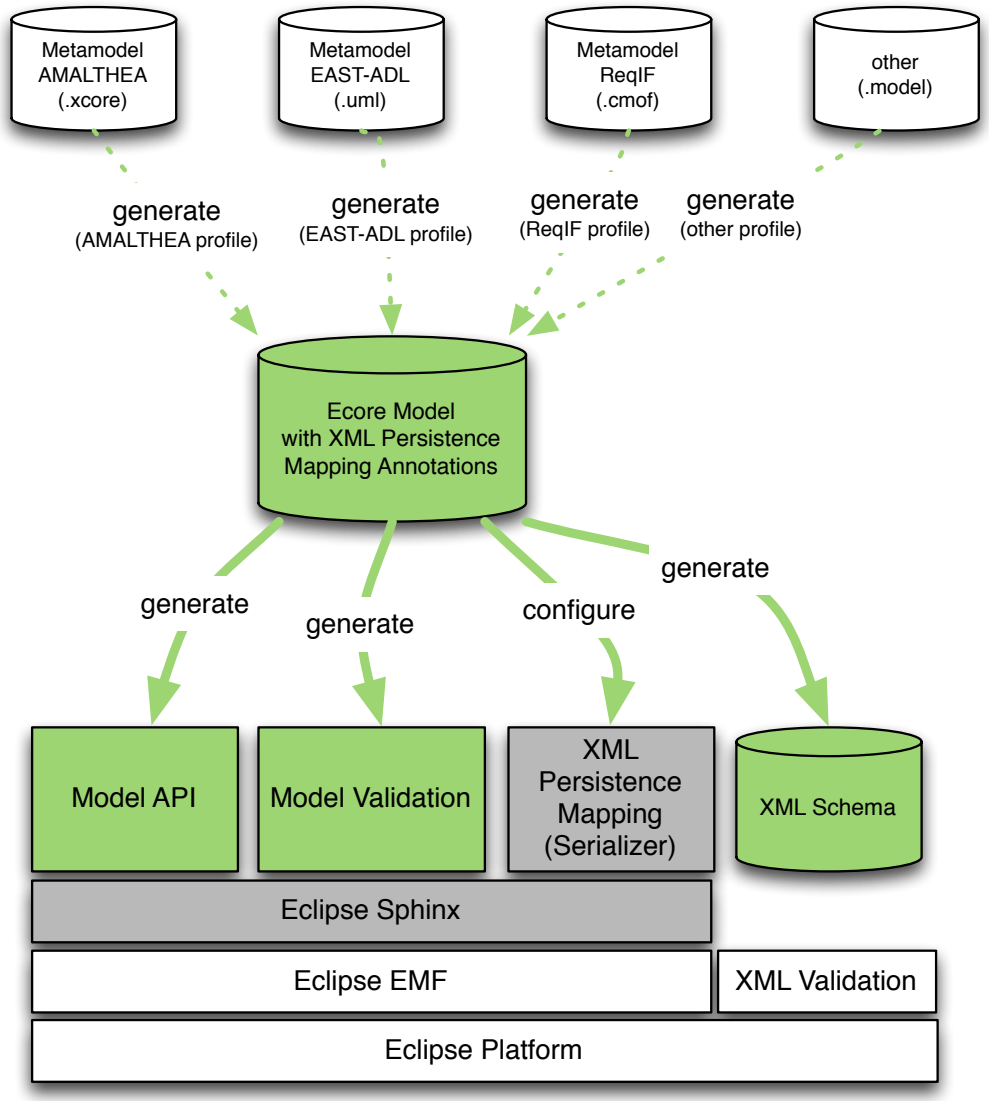


Figure 1: From Domain Model to Tool and XSD

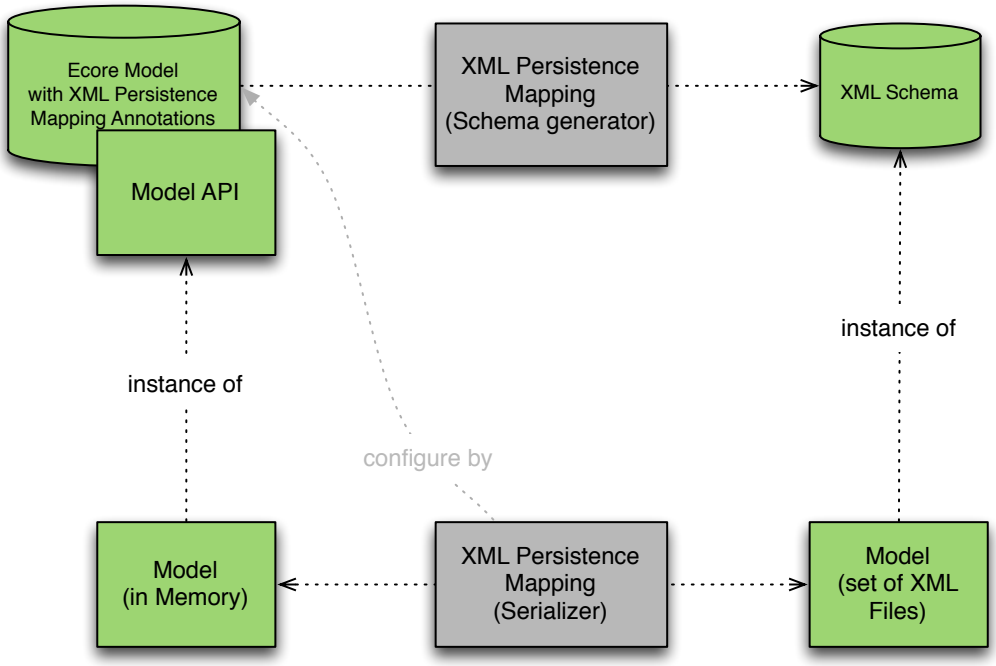


Figure 2: XML Persistence Mapping: Relation between XSD Schema generator and Serializer

- consistent naming conventions for mapping between the model and the XML-representation
- strict rules for usage of XML elements and XML attributes
- syntax completion by providing an XML schema to the XML editor

3.4 Validation

As an end-user I want to validate individual XML files against an XML Schema so that I can execute an initial quality check by means of generic XML Schema validators.

3.5 Text Diff

As an end-user I want to apply a textual diff tool on different versions of the XML files so that I can identify the model differences on file level. This requires deterministic and one-to-one persistence mapping between the model in memory and its XML representation.
[To do: move solution to section design principles] Patterns are:

- defined sequence of XML elements
- defined sequence of XML attributes
- preserve order of items in collections

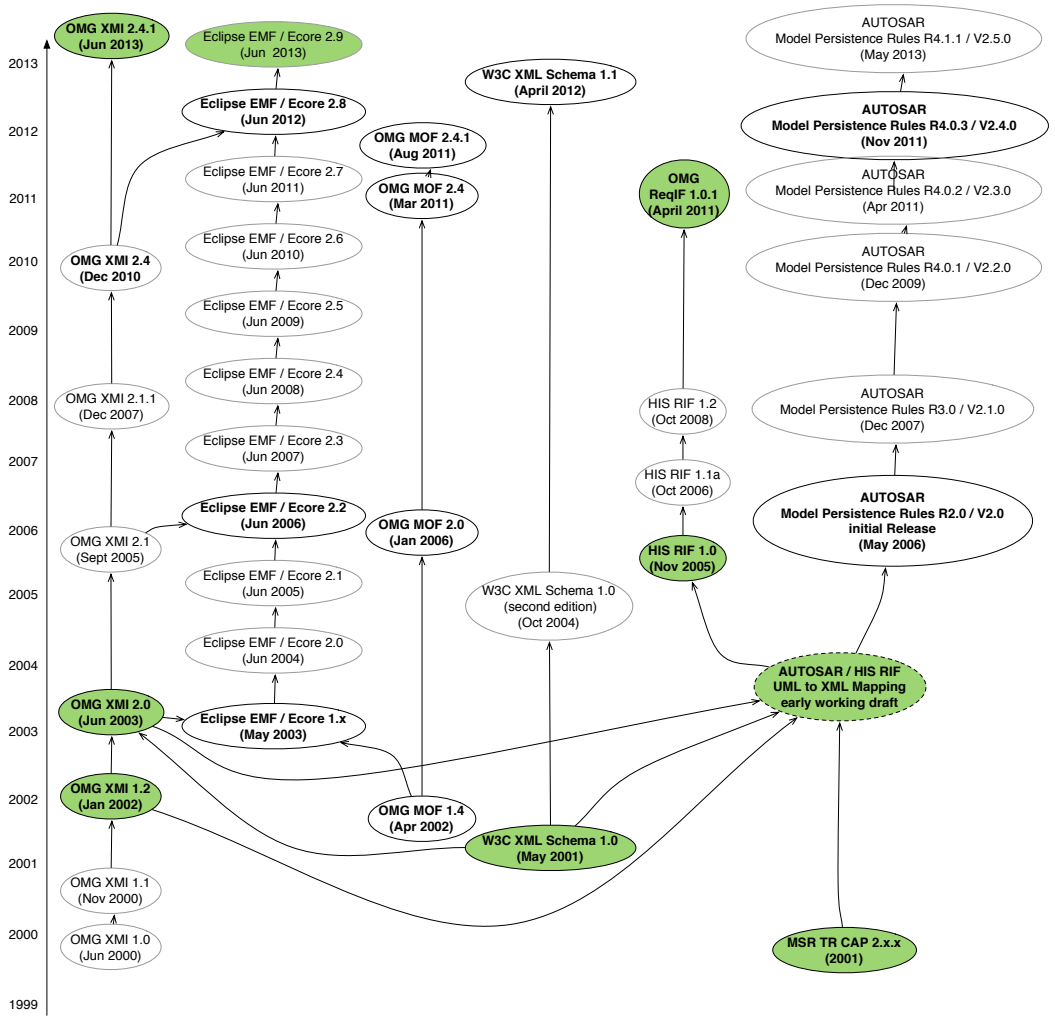
3.6 Full support of existing EMF persistence

As a tool developer I want to laverage all existing serialization features that are provided by the Eclipse EMF project so that I can still read and write models that do not explicitly support the XML persistence mapping extensions.

3.7 Model Evolution / Backwards compatibility

3.8 Manual annotation of metamodel

As a developer I want to efficiently be able to directly create and annotate my XCore or Ecore model as an input for the XML persistence mapping. E.g. I want to configure the default persistence mapping strategies at a central location so I do not need to add annotations to all model elements which might reduce the readability of my model.



4 Related Standards and Specifications

The following figure illustrates the relation and dependencies between the related standards:

4.1 OMG XMI

The OMG MOF 2 XMI Mapping specifications describe how to exchange metadata and models by means of XML files. The 2.x versions of the XMI standard are fundamentally different from the 1.x versions:

- XMI 1.x describes mapping rules between MOF and XML DTDs. Example of an UML model serialized using XMI 1.2 [4]:

```
<XMI xmi.version="1.2" xmlns:UML="org.omg/standards/UML">
  <XMI.content>
    <UML:Class name="ClassA">
      <UML:Classifier.feature>
        <UML:Attribute name="attributeA"
              visibility="private" />
      </UML:Classifier.feature>
    </UML:Class>
  </XMI.content>
</XMI>
```

- XMI 2.x describes mapping rules between MOF and XML Schema. These mappings produce more compact XML files. Example of an UML model serialized using XMI 2.4.1 [3]:

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701"
  xmlns:xmi="http://www.omg.org/spec/XMI/20110701">
  <uml:Class name="ClassA" xmi:type="uml:Class">
    <ownedAttribute xmi:type="uml:Property" name="attributeA"
      visibility="private" />
  </uml:Class>
</xmi:XMI>
```

4.2 OMG ReqIF

4.3 EMF XML Mapping

EMF has implemented XMI 2.x for serialization of models to XMI. Additionally, a more flexible mapping to XML schema is supported [1]. [To do: describe some configuration possibilities and advantages, e.g. feature maps]

5 Metamodel Best Practices

5.1 Ecore Patterns

5.1.1 Use primitive types when possible

Ecore provides several predefined data types. Whenever possible, the datatype that maps to a primitive Java type should be used. E.g. a boolean values can be represented by EBoolean (java.lang.boolean) or EBooleanObject (java.lang.Boolean). java.lang.Boolean wraps the primitive type java.lang.boolean and thus adds some overhead with respect to resource consumption. Additionally, wrapping the primitive type into an object adds the additional value 'null' which needs to be handled in your application.

5.1.2 Explicitly mark features unsetable

By explicitly setting EStructuralFeatures to unsetable, the EMF Code generator adds flags and and an API that allows tracking if the EStructuralFeature is set. If the unsetable option is not set, EMF derives that information from the value of the feature: If the value equals to its default value or if a list is empty then it is considered as unset. Thus, it is not possible to distinguish between a value that was explicitly set to the default value and a value that was not set at all.
[To do: add note on additonal memory consumption and how to overcome it]

5.1.3 Explicitly mark features ordered

EStructuralFeatures that represent multiple values (isMany()==true) can have the semantics of an ordered list or an unordered collection. The semantics of the ordered list should be preferred for the following reasons:

- preserve the order of elements when reading and writing models (important for textual diff of XML files)
- deterministic navigation in model is prerequisit for deterministic code generation

Note: The EMF code generator currently ignores the 'ordered' property and always generates ELists. However, this might change in future and frameworks such as QVTO or OCL already evaluate the flag.

5.1.4 Avoid sub packages

Unless the class names in the model are not unique, nested packages are unnecessary syntactic sugar and often end up causing problems because some aspect of the framework doesn't work perfectly for them.

5.2 EMF Code Generator

5.3 Java Performance Tuning

Performance tuning in Java is highly related to the implementation of the Java Virtual Machine. Please see [2] for more details. Additional performance optimizations are described in [5].[2]

6 XML Best Practices

6.1 XML Schema Patterns

6.1.1 Use Schema for Validation only

XML Schema can be used for validation of XML documents as well as a provider of default values. If default values are defined in the XML Schema the XML processor might produce different values if the document is pared with or without availability of the XML Schema. Thus, the XML schema should be used for validation only and no default values should be defined.

6.1.2 Prefer xsd:sequence

XML complex types can group the contained XML elements in sequential (xsd:sequence), disjunctive (xsd:choice) or conjunctive (xsd:all) manner. Whenever applicable, the xsd:sequence pattern schould be preferred since it enforces a given sequence of XML elements in the XML document which simplifies textual diff of XML files.

6.1.3 Avoid extension or restriction of xsd:complexTypes

[To do: provide more background information: only single inheritance, requires repitition of elements, complex to validate. discuss named xsd:group vs. repitition of elements in each complex type]

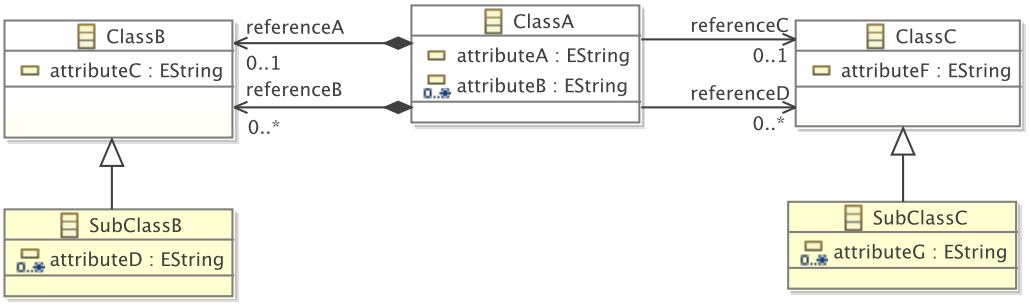
6.2 XML Document Patterns

7 Ecore to XML Mapping Design Principles

[To do: principles]

7.1 Overview

[To do: what do we need for serialization without loss of information]



7.2 Customization for Different Meta Models

[To do: todo dependent on primary usecases
two step approach
Language specific profiles (mapping of annotated UML to fully annotated XML Persistence Mapping Model)
Rules described in this document
]

7.3 XML Elements vs. Attributes

[To do: XML Elements vs. Attributes]

7.4 XML Version

[To do: Impact Table: Metamodel / Model / XML Schema / Schema]

7.5 Linking

[To do: Linking]

7.6 Null and Default Values

[To do: Null and Default Values]

7.7 Primitive Types

[To do: primitive types: xsd:types vs. refined xsd:strings]

7.8 Packages and Namespaces

[To do: Packages and Namespaces]

7.9 EAttributes with occurance bigger than 1

Each XML attribute may occur only once in the context of an XML. For mapping of EAttribute that represent a list of values, two alternatives are possible:

1. Representation by a single XML attribute or XML element which carries the list of values separated by a special character.
2. Representation by multiple XML elements. Each wraps an individual value.

7.10 Extensions

[To do: AUTOSAR: splittable
other concepts]

7.11 Inheritance

[To do: xmi:type vs. xsi:type vs. schema inheritance and restriction]

7.12 Validation

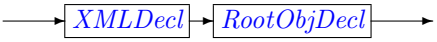
[To do: levels of validation]

8 XML Document Production

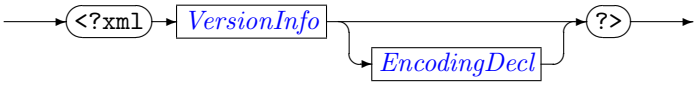
This Chapter contains detailed information for implementers of tools that read and write XML documents according to the XML Persistence Mappings.

EBNF - Syntax Diagram

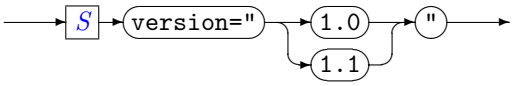
document



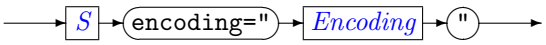
XMLDecl



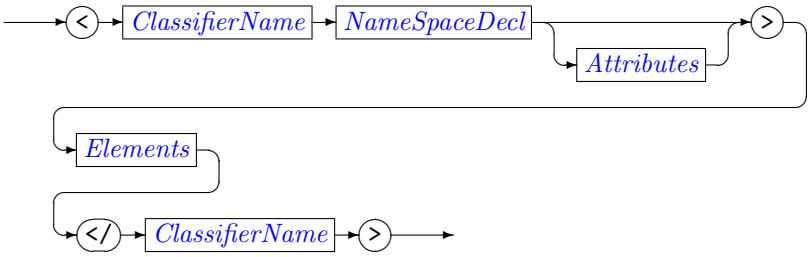
VersionInfo



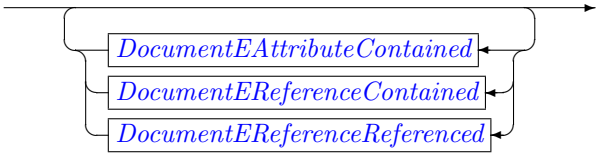
EncodingDecl



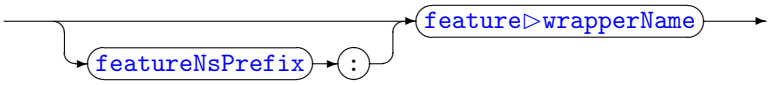
RootObjDecl



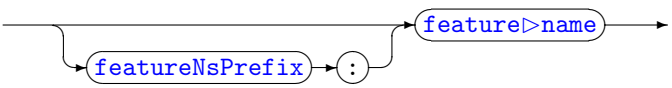
Elements



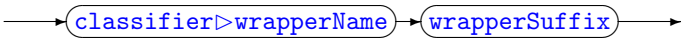
QFeatureWrapperName



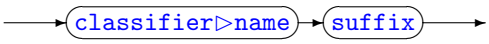
QFeatureName



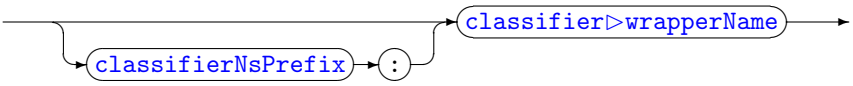
ExtClassifierWrapperName



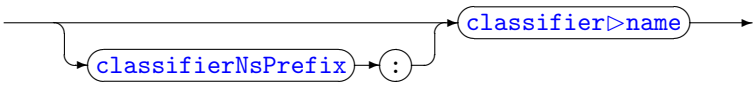
ExtClassifierName



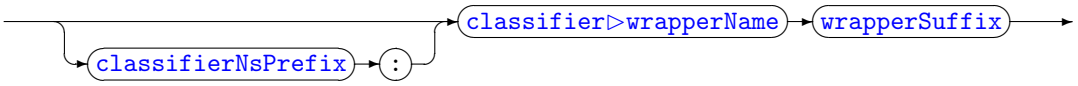
QClassifierWrapperName



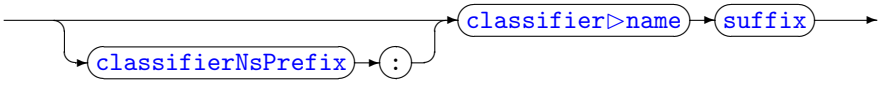
QClasifierName



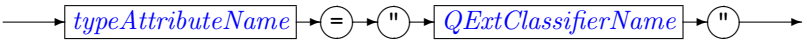
QExtClassifierWrapperName



QExtClasifierName



TypeAttribute



8.1 Document Mapping for EAttributeAttribute

EBNF - Syntax Diagram

Document Example

8.2 Document Mappings for EAttributeContained

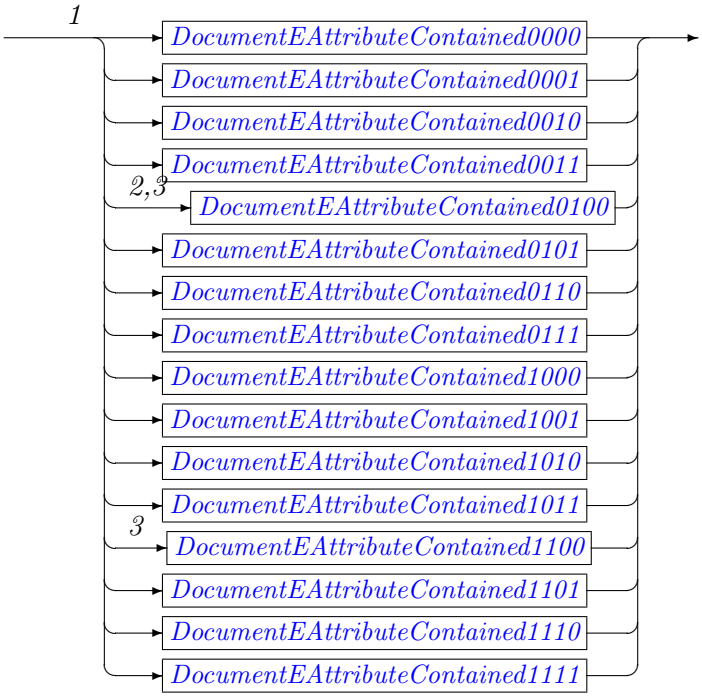
This rule maps the EAttribute to an XML element.

Preconditions EAttributeContainedxxxx mappings are applicable if

$\wedge\, x\ instanceof$ EAttribute
$\wedge\, false == x.isTransient()$
$\wedge\, ExtendedMetaData.ELEMENT\_FEATURE == extendedMetaData.getFeatureKind(x)$

EBNF - Syntax Diagram

DocumentEAttributeContained



Constraints and Parameterization

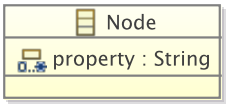
1. Selection of DocumentEAttributeContainedxxxx Rules:

	XMLPersistenceExtendeMetaData			
Mapping Rule ...EAttribute...	feature WrapperElement	feature Element	classifier WrapperElement	classifier Element
not applicable	false	false	false	false
...Contained0001	false	false	false	true
...Contained0010	false	false	true	false
...Contained0011	false	false	true	true
...Contained0100	false	true	false	false
...Contained0101	false	true	false	true
...Contained0110	false	true	true	false
...Contained0111	false	true	true	true
...Contained1000	true	false	false	false
...Contained1001	true	false	false	true
...Contained1010	true	false	true	false
...Contained1011	true	false	true	true
...Contained1100	true	true	false	false
...Contained1101	true	true	false	true
...Contained1110	true	true	true	false
...Contained1111	true	true	true	true

2. If XMLPersistenceExtendedMetaData is not defined, then rule DocumentEAttributeContained0100 applies (default EMF persistence behaviour).

3. If XMLPersistenceExtendedMetaData is defined and one of the keys featureWrapperElement, featureElement, classifierWrapperElement, classifierWrapper is missing, then the rule DocumentEAttributeContained1100 applies for EAttributes with $true == x.isMany()$ and rule DocumentEAttributeContained0100 otherwise. Since these missing keys identify an error in the configuration, the (de)serializer shall log an error message.

Description of Example The EAttributeContained examples are based on the following minimal ecore model:




```
/**
 * Class Node
 */
@ExtendedMetaData(name="NODE")
@XMLPersistenceMappingExtendedMetaData(wrapperName="NODE")
class Node {
        @ExtendedMetaData(name="NODE")
        @XMLPersistenceMappingExtendedMetaData(
```

```
                wrapperName="NODES",
                featureWrapperElement="true",
                featureElement="true",
                classifierWrapperElement="false",
                classifierElement="false"
        )
        String[] property
}


/**
 * Datatype String
 */
@ExtendedMetaData(name="STRING")
@XMLPersistenceMappingExtendedMetaData(wrapperName="STRING-VALUES")
type String wraps java.lang.String
```

The model consists of a single 'Node' Object that has two values for the EAttribute 'property': 'Property1' and 'Property2'

8.2.1 Document Mapping for EAttributeContained0000

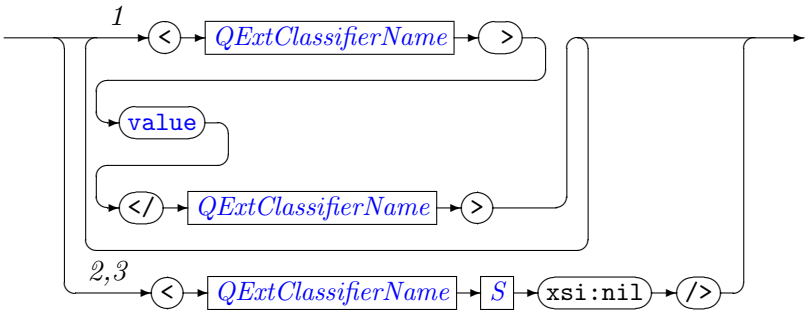
Invalid invalid mapping rule.

8.2.2 Document Mapping for EAttributeContained0001

Preconditions [To do: todo]

EBNF - Syntax Diagram

DocumentEAttributeContained0001



Constraints and Parameterization

1. If $\neg(obj.eGet(x).isEmpty() \vee null = obj.eGet(x))$ then create start and end XML elements that contain the value

2. If $true = x.isMany() \wedge obj.eGet(x).isEmpty()$ then xsi:nil shall be used.
3. If $false = x.isMany() \wedge true = obj.eIsSet(x) \wedge null = obj.eGet()$ then xsi:nil shall be used.

[To do: do we need to handle null values in lists?]:

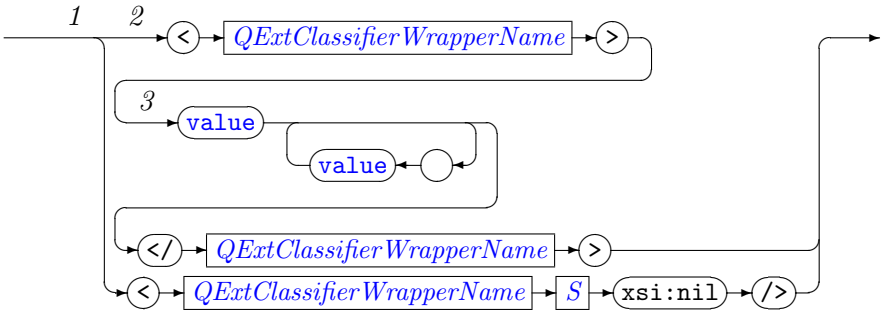
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <STRING>Property1</STRING>
  <STRING>Property2</STRING>
</NODE>
```

8.2.3 Document Mapping for EAttributeContained0010

EBNF - Syntax Diagram

DocumentEAttributeContained0010



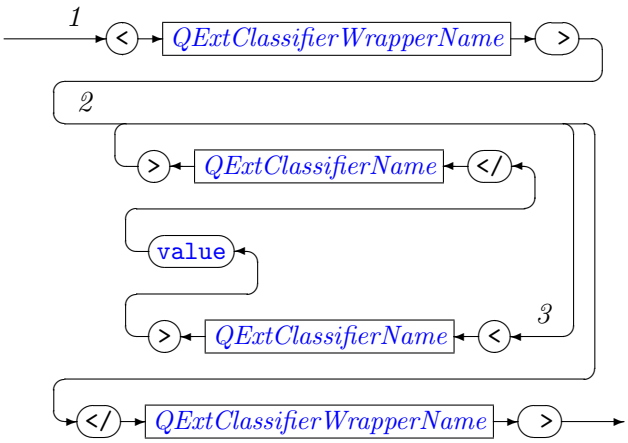
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <STRING-VALUES>Property1 Property2</STRING-VALUES>
</NODE>
```

8.2.4 Document Mapping for EAttributeContained0011

EBNF - Syntax Diagram

DocumentEAttributeContained0011



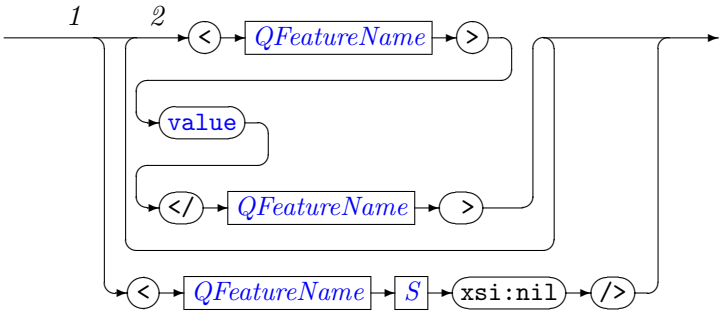
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <STRING-VALUES>
    <STRING>Property1</STRING>
    <STRING>Property2</STRING>
  </STRING-VALUES>
</NODE>
```

8.2.5 Document Mapping for EAttributeContained0100

EBNF - Syntax Diagram

DocumentEAttributeContained0100



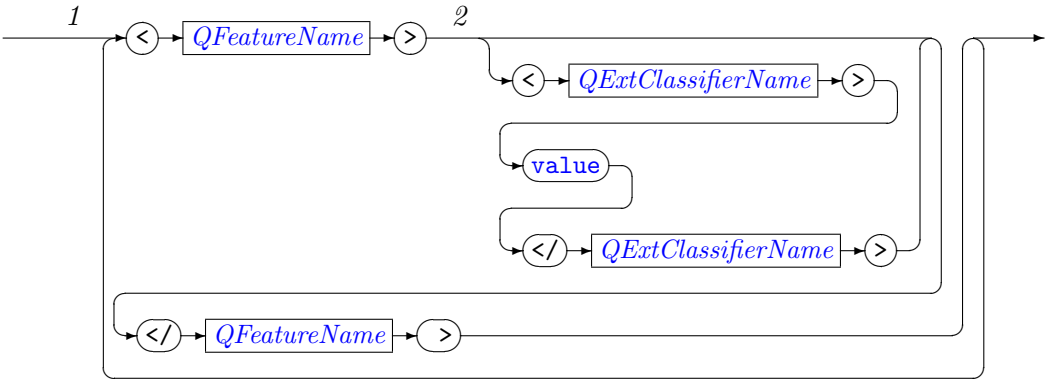
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTY>Property1</PROPERTY>
  <PROPERTY>Property2</PROPERTY>
</NODE>
```

8.2.6 Document Mapping for EAttributeContained0101

EBNF - Syntax Diagram

DocumentEAttributeContained0101



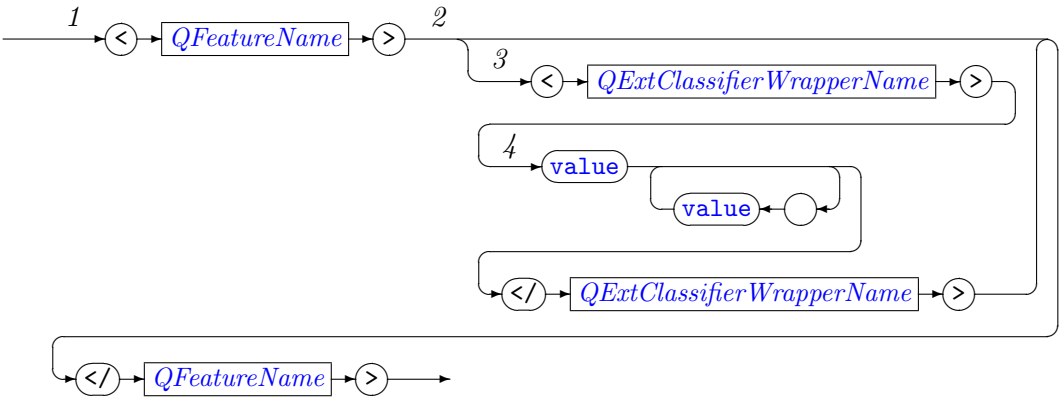
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTY>
    <STRING>Property1</STRING>
  </PROPERTY>
  <PROPERTY>
    <STRING>Property2</STRING>
  </PROPERTY>
</NODE>
```

8.2.7 Document Mapping for EAttributeContained0110

EBNF - Syntax Diagram

DocumentEAttributeContained0110



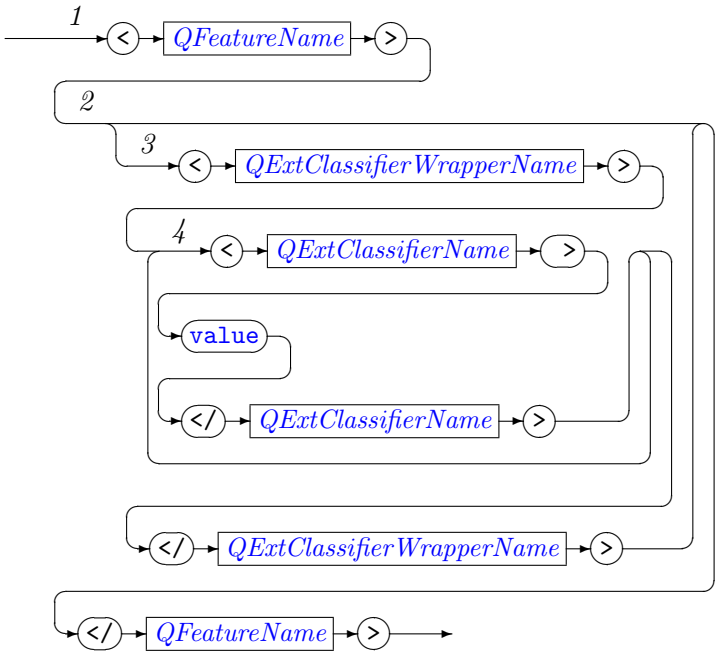
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTY>
    <STRING-VALUES>Property1 Property2</STRING-VALUES>
  </Property>
</NODE>
```

8.2.8 Document Mapping for EAttributeContained0111

EBNF - Syntax Diagram

DocumentEAttributeContained0111



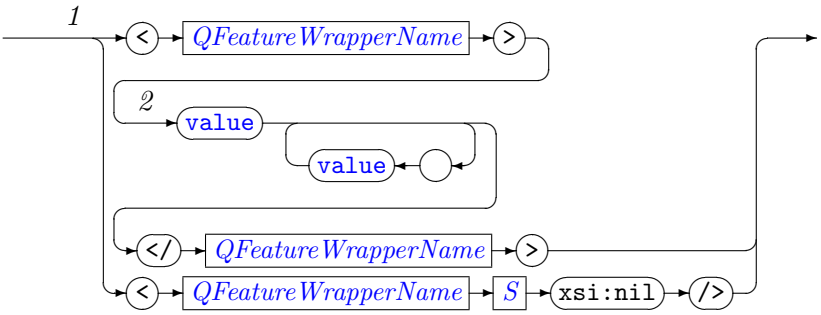
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTY>
    <STRING-VALUES>
      <STRING>Property1</STRING>
      <STRING>Property2</STRING>
    </STRING-VALUES>
  </PROPERTY>
</NODE>
```

8.2.9 Document Mapping for EAttributeContained1000

EBNF - Syntax Diagram

DocumentEAttributeContained1000



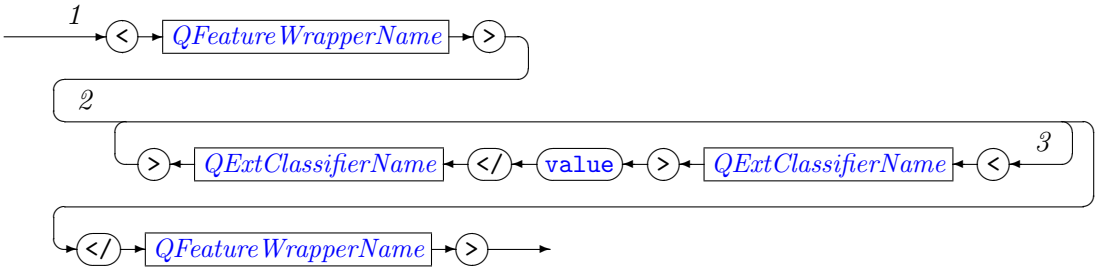
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>Property1 Property2</PROPERTIES>
</NODE>
```

8.2.10 Document Mapping for EAttributeContained1001

EBNF - Syntax Diagram

DocumentEAttributeContained1001



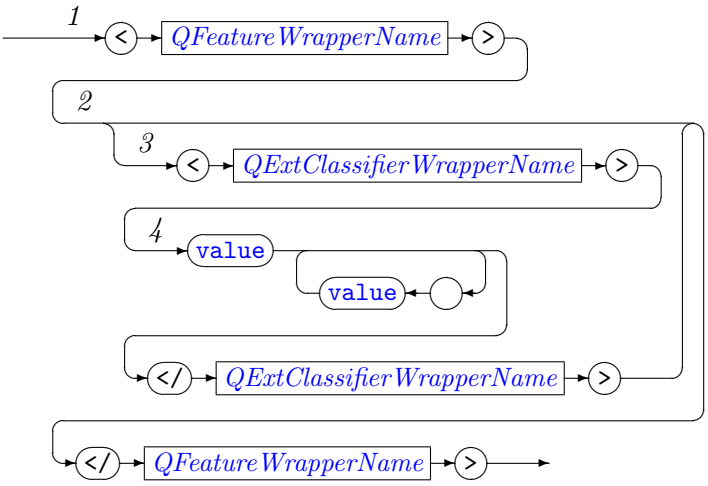
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <STRING>Property1</STRING>
    <STRING>Property2</STRING>
  </PROPERTIES>
</NODE>
```

8.2.11 Document Mapping for EAttributeContained1010

EBNF - Syntax Diagram

DocumentEAttributeContained1010



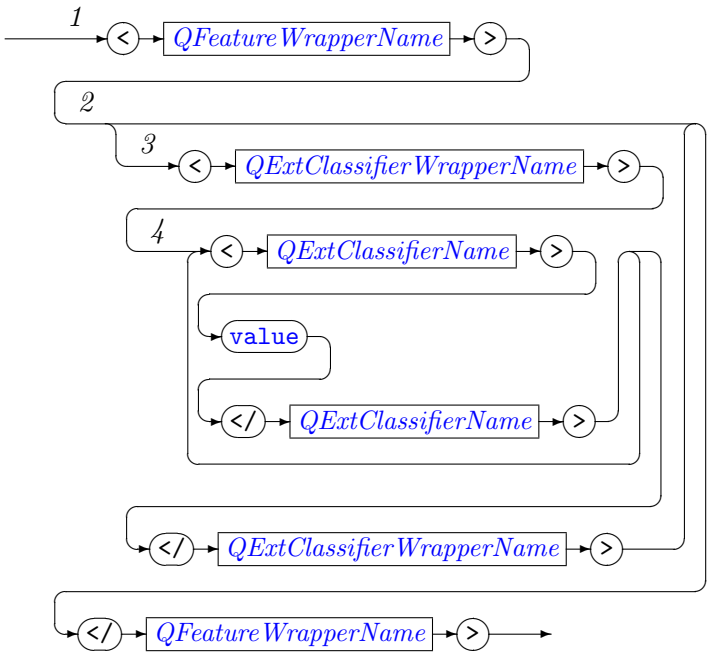
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <STRING-VALUES>Property1 Property2</STRING-VALUES>
  </PROPERTIES>
</NODE>
```

8.2.12 Document Mapping for EAttributeContained1011

EBNF - Syntax Diagram

DocumentEAttributeContained1011



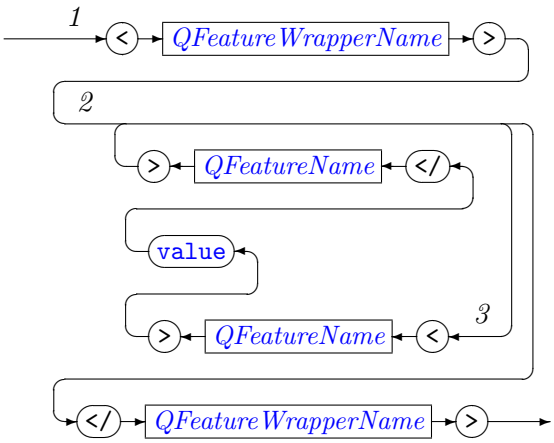
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <STRING-VALUES>
      <STRING>Property1</STRING>
      <STRING>Property2</STRING>
    </STRING-VALUES>
  </PROPERTIES>
</NODE>
```

8.2.13 Document Mapping for EAttributeContained1100

EBNF - Syntax Diagram

DocumentEAttributeContained1100



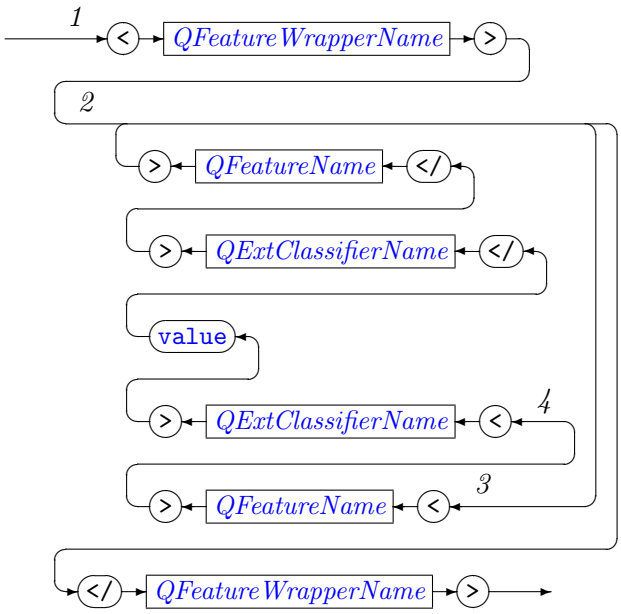
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <PROPERTY>Property1</PROPERTY>
    <PROPERTY>Property2</PROPERTY>
  </PROPERTIES>
</NODE>
```

8.2.14 Document Mapping for EAttributeContained1101

EBNF - Syntax Diagram

DocumentEAttributeContained1101



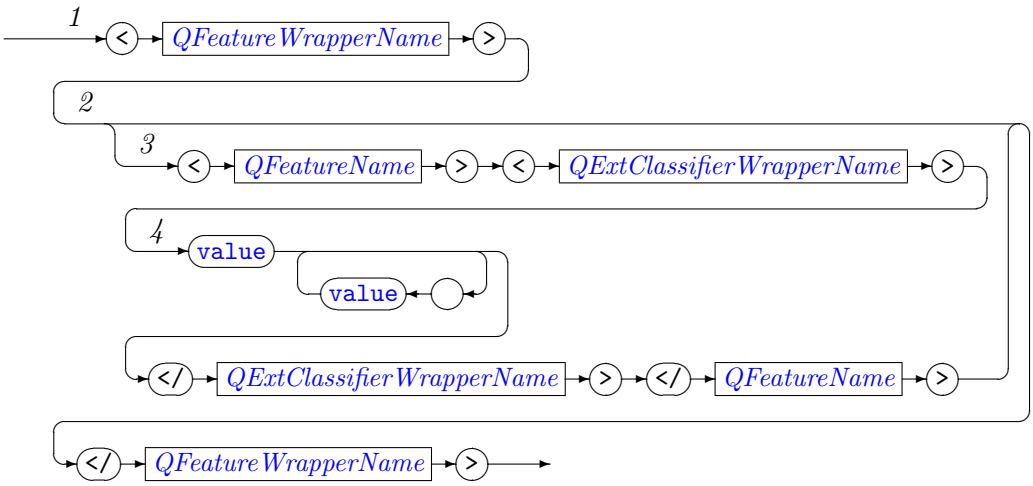
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <PROPERTY>
      <STRING>Property1</STRING>
    </PROPERTY>
    <PROPERTY>
      <STRING>Property2</STRING>
    </PROPERTY>
  </PROPERTIES>
</NODE>
```

8.2.15 Document Mapping for EAttributeContained1110

EBNF - Syntax Diagram

DocumentEAttributeContained1110



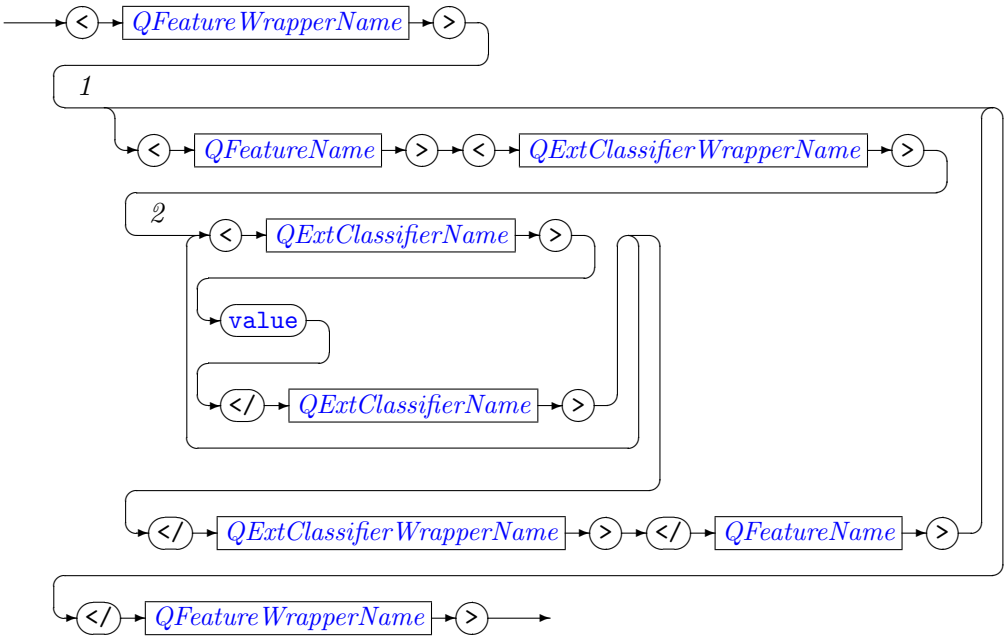
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <PROPERTY>
      <STRING-VALUES>Property1 Property2</STRING-VALUES>
    </PROPERTY>
  </PROPERTY>
</NODE>
```

8.2.16 Document Mapping for EAttributeContained1111

EBNF - Syntax Diagram

DocumentEAttributeContained1111



Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI">
  <PROPERTIES>
    <PROPERTY>
      <STRING-VALUES>
        <STRING>Property1</STRING>
        <STRING>Property2</STRING>
      </STRING-VALUES>
    </PROPERTY>
  </PROPERTIES>
</NODE>
```

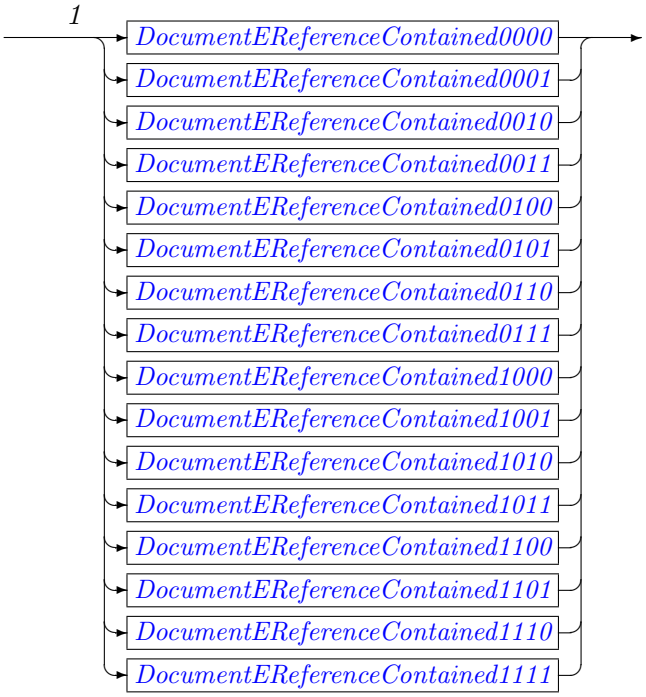
8.3 Document Mappings for containment EReferences

Preconditions Document mapping is applicable if

$$
\begin{aligned}
& false == x.isTransient() \\
\wedge\, & true == x.isContainment() \\
\wedge\, & ExtendedMetaData.ELEMENT\_FEATURE == extendedMetaData.getFeatureKind(x)
\end{aligned}
$$

EBNF - Syntax Diagram

DocumentEReferenceContained



Note 1

	XMLPersistenceExtendeMetaData			
Mapping Rule	feature WrapperElement	feature Element	classifier WrapperElement	classifier Element
...EReferenceContained0000	false	false	false	false
...EReferenceContained0001	false	false	false	true
...EReferenceContained0010	false	false	true	false
...EReferenceContained0011	false	false	true	true
...EReferenceContained0100	false	true	false	false
...EReferenceContained0101	false	true	false	true
...EReferenceContained0110	false	true	true	false
...EReferenceContained0111	false	true	true	true
...EReferenceContained1000	true	false	false	false
...EReferenceContained1001	true	false	false	true
...EReferenceContained1010	true	false	true	false
...EReferenceContained1011	true	false	true	true
...EReferenceContained1100	true	true	false	false
...EReferenceContained1101	true	true	false	true
...EReferenceContained1110	true	true	true	false
...EReferenceContained1111	true	true	true	true

The following default mapping rules apply:

- If XMLPersistenceExtendedMetaData is not defined, then rule DocumentEReferenceContained0100 applies (default EMF persistence behaviour).
- If XMLPersistenceExtendedMetaData is defined and one of the keys featureWrapperElement, featureElement, classifierWrapperElement, classifierWrapper is missing, then the rule DocumentEReferenceContained1001 applies for EReferences with $true == x.isMany()$ and rule DocumentEReferenceContained0101 otherwise. Since these missing keys identify an error in the configuration, the (de)serializer shall log an error message.

8.3.1 Document Mapping for EReferenceContained0000

[To do: todo]

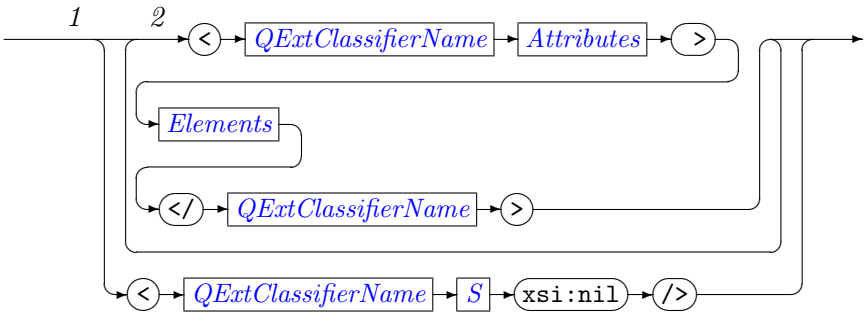
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <DESC>this is childNode 1</DESC>
  <DESC>this is childNode 2</DESC>
  <DESC>this is childSubNode</DESC>
</NODE>
```

8.3.2 Document Mapping for EReferenceContained0001

EBNF - Syntax Diagram

DocumentEReferenceContained0001



Document Example

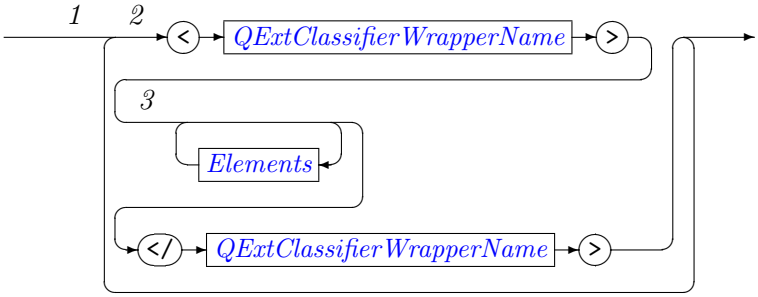
```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <NODE name="childNode1">
    <DESC>this is childNode 1</DESC>
```

```
  </NODE>
  <NODE name="childNode2">
    <DESC>this is childNode 2</DESC>
  </NODE>
  <SUB-NODE name="childSubNode">
    <DESC>this is childSubNode</DESC>
  </SUB-NODE>
</NODE>
```

8.3.3 Document Mapping for EReferenceContained0010

EBNF - Syntax Diagram

DocumentEReferenceContained0010



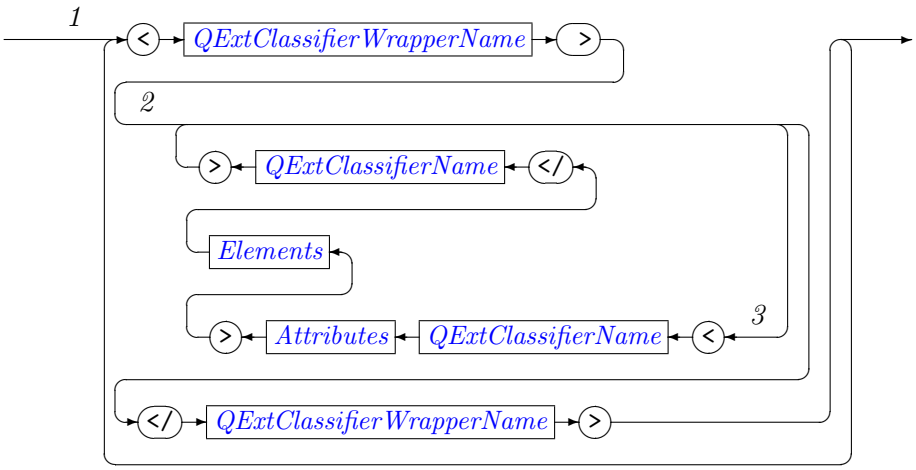
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <NODES>
    <DESC>this is childNode 1</DESC>
    <DESC>this is childNode 2</DESC>
  </NODES>
  <SUB-NODES>
    <DESC>this is childSubNode</DESC>
  </SUB-NODES>
</NODE>
```

8.3.4 Document Mapping for EReferenceContained0011

EBNF - Syntax Diagram

DocumentEReferenceContained0011



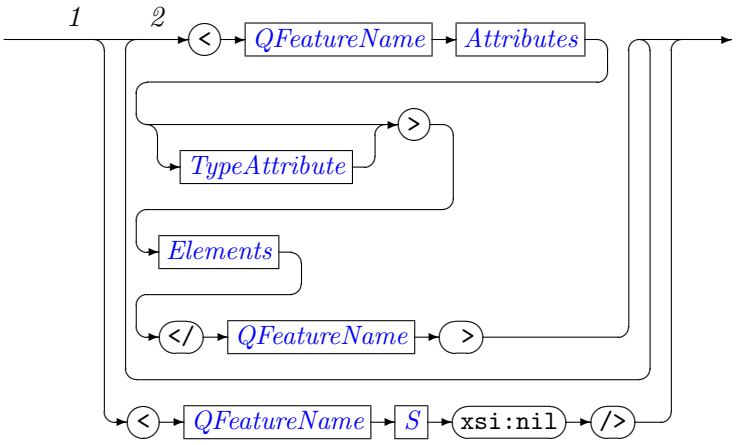
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <NODES>
    <NODE name="childNode1">
      <DESC>this is childNode 1</DESC>
    </NODE>
    <NODE name="childNode2">
      <DESC>this is childNode 2</DESC>
    </NODE>
  </NODES>
  <SUB-NODES>
    <SUB-NODE name="childSubNode">
      <DESC>this is childSubNode</DESC>
    </SUB-NODE>
  </SUB-NODES>
</NODE>
```

8.3.5 Document Mapping for EReferenceContained0100

EBNF - Syntax Diagram

DocumentEReferenceContained0100



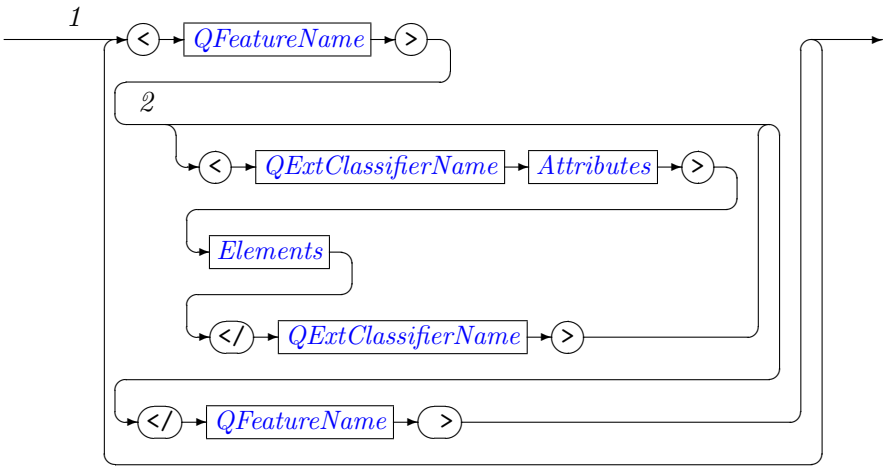
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="NodesURI" name="root">
  <CHILD name="childNode1">
    <DESC>this is childNode 1</DESC>
  </CHILD>
  <CHILD name="childNode2">
    <DESC>this is childNode 2</DESC>
  </CHILD>
  <CHILD TYPE="SUB-NODE" name="childSubNode">
    <DESC>this is childSubNode</DESC>
  </CHILD>
</NODE>
```

8.3.6 Document Mapping for EReferenceContained0101

EBNF - Syntax Diagram

DocumentEReferencedContained0101



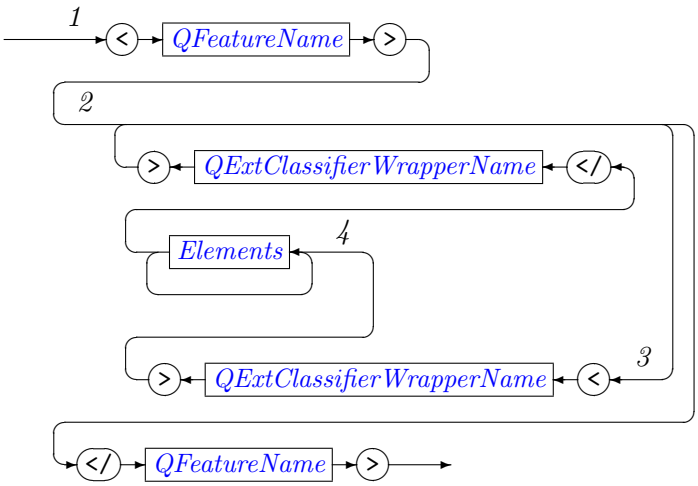
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILD>
    <NODE name="childNode1">
      <DESC>this is childNode 1</DESC>
    </NODE>
  </CHILD>
  <CHILD>
    <NODE name="childNode2">
      <DESC>this is childNode 2</DESC>
    </NODE>
  </CHILD>
  <CHILD>
    <SUB-NODE name="childSubNode">
      <DESC>this is childSubNode</DESC>
    </SUB-NODE>
  </CHILD>
</NODE>
```

8.3.7 Document Mapping for EReferenceContained0110

EBNF - Syntax Diagram

DocumentEReferenceContained0110



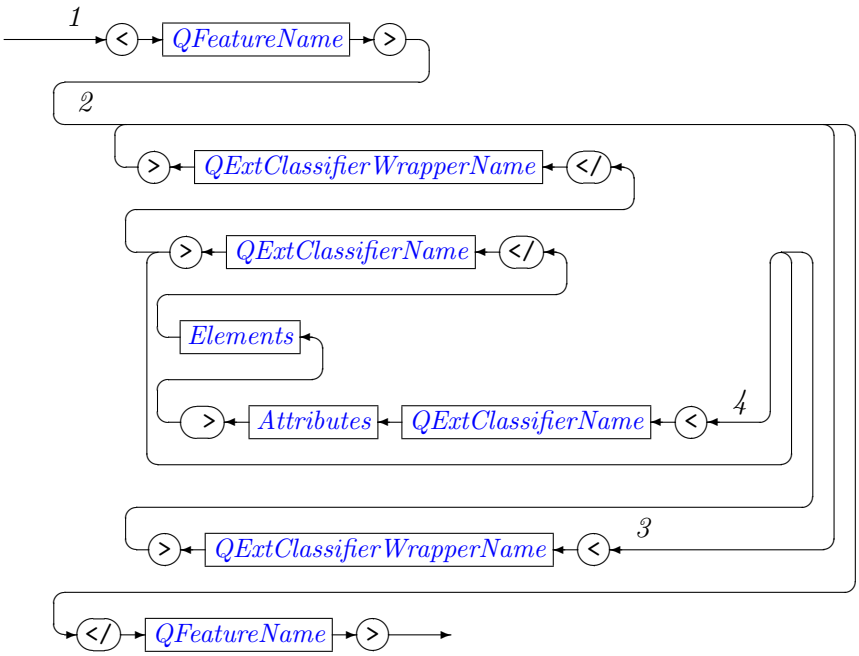
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILD>
    <NODES>
      <DESC>this is childNode 1</DESC>
      <DESC>this is childNode 2</DESC>
    </NODES>
    <SUB-NODES>
      <DESC>this is childSubNode</DESC>
    </SUB-NODES>
  </CHILD>
</NODE>
```

8.3.8 Document Mapping for EReferenceContained0111

EBNF - Syntax Diagram

DocumentEReferenceContained0111



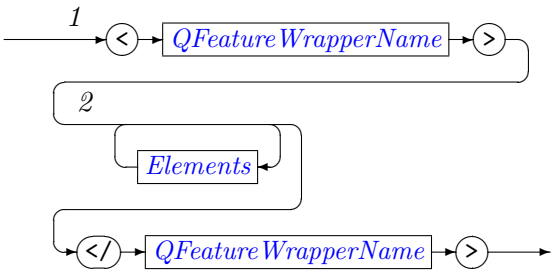
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILD>
    <NODES>
      <NODE name="childNode1">
        <DESC>this is childNode 1</DESC>
      </NODE>
      <NODE name="childNode2">
        <DESC>this is childNode 2</DESC>
      </NODE>
    </NODES>
    <SUB-NODES>
      <SUB-NODE name="childSubNode">
        <DESC>this is childSubNode</DESC>
      </SUB-NODE>
    </SUB-NODES>
  </CHILD>
</NODE>
```

8.3.9 Document Mapping for EReferenceContained1000

EBNF - Syntax Diagram

DocumentEReferenceContained1000



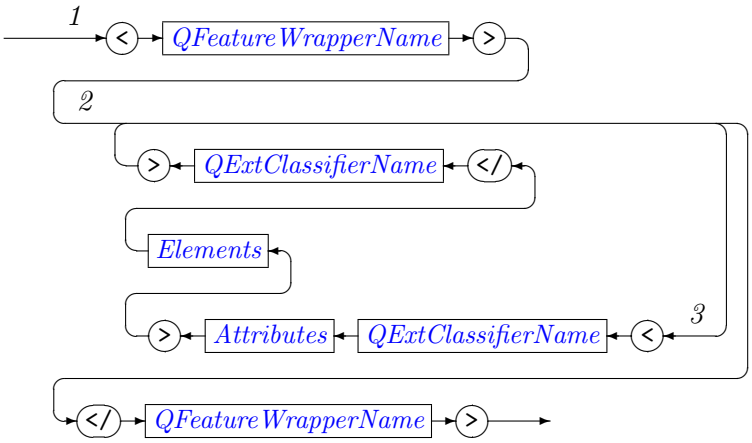
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <DESC>this is childNode 1</DESC>
    <DESC>this is childNode 2</DESC>
    <DESC>this is childSubNode</DESC>
  </CHILDREN>
</NODE>
```

8.3.10 Document Mapping for EReferenceContained1001

EBNF - Syntax Diagram

DocumentEReferenceContained1001



Document Example

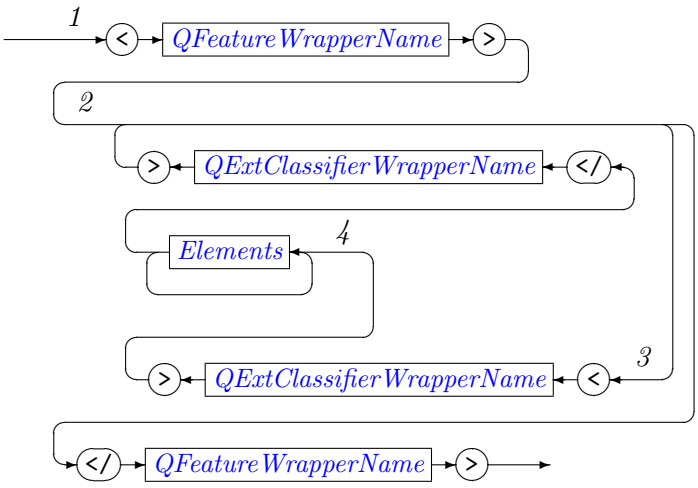
```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="childNode1">
      <DESC>this is childNode 1</DESC>
```

```
    </NODE>
    <NODE name="childNode2">
      <DESC>this is childNode 2</DESC>
    </NODE>
    <SUB-NODE name="childSubNode">
      <DESC>this is childSubNode</DESC>
    </SUB-NODE>
  </CHILDREN>
</NODE>
```

8.3.11 Document Mapping for EReferenceContained1010

EBNF - Syntax Diagram

DocumentEReferenceContained1010



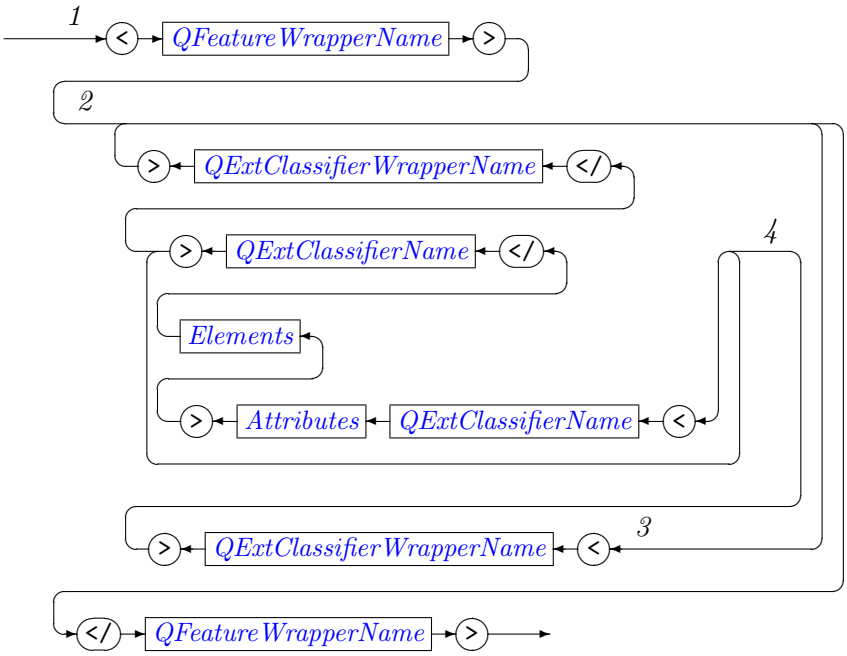
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODES>
      <DESC>this is childNode 1</DESC>
      <DESC>this is childNode 2</DESC>
    </NODES>
    <SUB-NODES>
      <DESC>this is childSubNode</DESC>
    </SUB-NODES>
  </CHILDREN>
</NODE>
```

8.3.12 Document Mapping for EReferenceContained1011

EBNF - Syntax Diagram

DocumentEReferenceContained1011



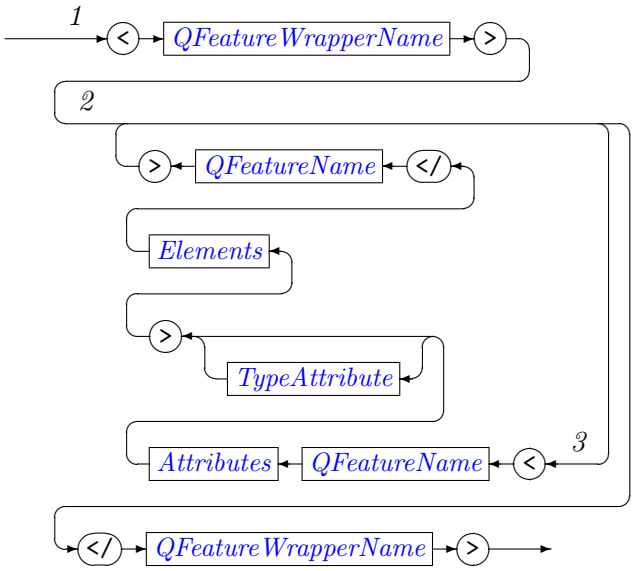
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODES>
      <NODE name="childNode1">
        <DESC>this is childNode 1</DESC>
      </NODE>
      <NODE name="childNode2">
        <DESC>this is childNode 2</DESC>
      </NODE>
    </NODES>
    <SUB-NODES>
      <SUB-NODE name="childSubNode">
        <DESC>this is childSubNode</DESC>
      </SUB-NODE>
    </SUB-NODES>
  </CHILDREN>
</NODE>
```

8.3.13 Document Mapping for EReferenceContained1100

EBNF - Syntax Diagram

DocumentEReferenceContained1100



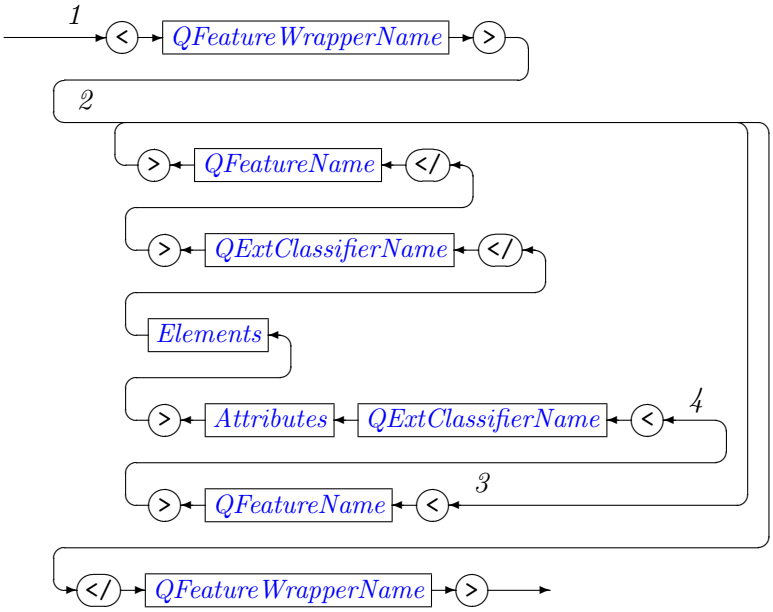
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="NodesURI" name="root">
  <CHILDREN>
    <CHILD name="childNode1">
      <DESC>this is childNode 1</DESC>
    </CHILD>
    <CHILD name="childNode2">
      <DESC>this is childNode 2</DESC>
    </CHILD>
    <CHILD TYPE="SUB-NODE" name="childSubNode">
      <DESC>this is childSubNode</DESC>
    </CHILD>
  </CHILDREN>
</NODE>
```

8.3.14 Document Mapping for EReferenceContained1101

EBNF - Syntax Diagram

DocumentEReferenceContained1101



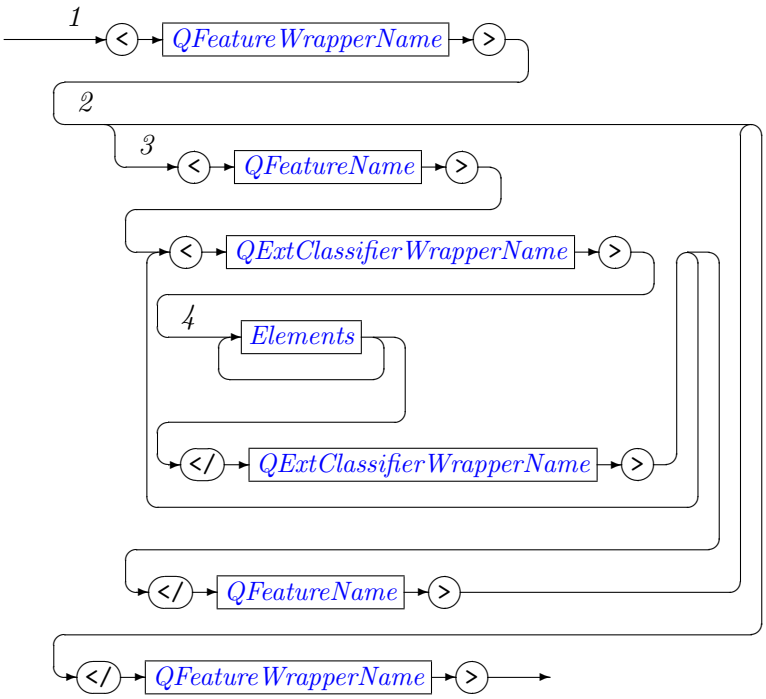
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <CHILD>
      <NODE name="childNode1">
        <DESC>this is childNode 1</DESC>
      </NODE>
    </CHILD>
    <CHILD>
      <NODE name="childNode2">
        <DESC>this is childNode 2</DESC>
      </NODE>
    </CHILD>
    <CHILD>
      <SUB-NODE name="childSubNode">
        <DESC>this is childSubNode</DESC>
      </SUB-NODE>
    </CHILD>
  </CHILDREN>
</NODE>
```

8.3.15 Document Mapping for EReferenceContained1110

EBNF - Syntax Diagram

DocumentEReferenceContained1110



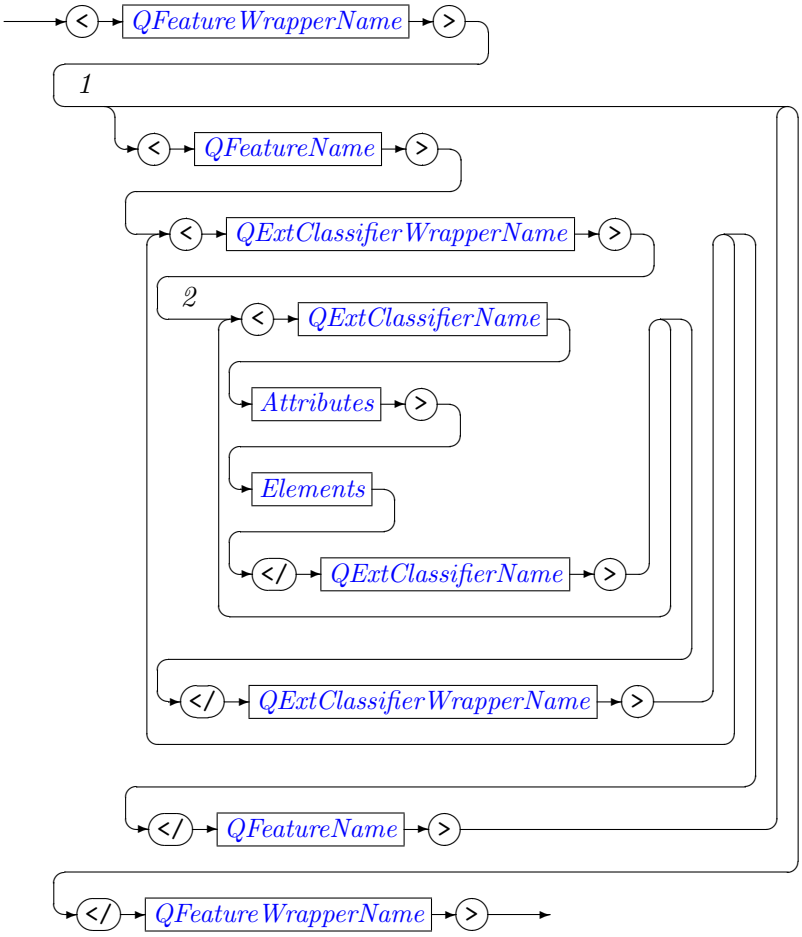
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <CHILD>
      <NODES>
        <DESC>this is childNode 1</DESC>
        <DESC>this is childNode 2</DESC>
      </NODES>
      <SUB-NODES>
        <DESC>this is childSubNode</DESC>
      </SUB-NODES>
    </CHILD>
  </CHILDREN>
</NODE>
```

8.3.16 Document Mapping for EReferenceContained1111

EBNF - Syntax Diagram

DocumentEReferenceContained1111



Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <CHILD>
      <NODES>
        <NODE name="childNode1">
          <DESC>this is childNode 1</DESC>
        </NODE>
        <NODE name="childNode2">
          <DESC>this is childNode 2</DESC>
        </NODE>
      </NODES>
      <SUB-NODES>
        <SUB-NODE name="childSubNode">
          <DESC>this is childSubNode</DESC>
        </SUB-NODE>
```

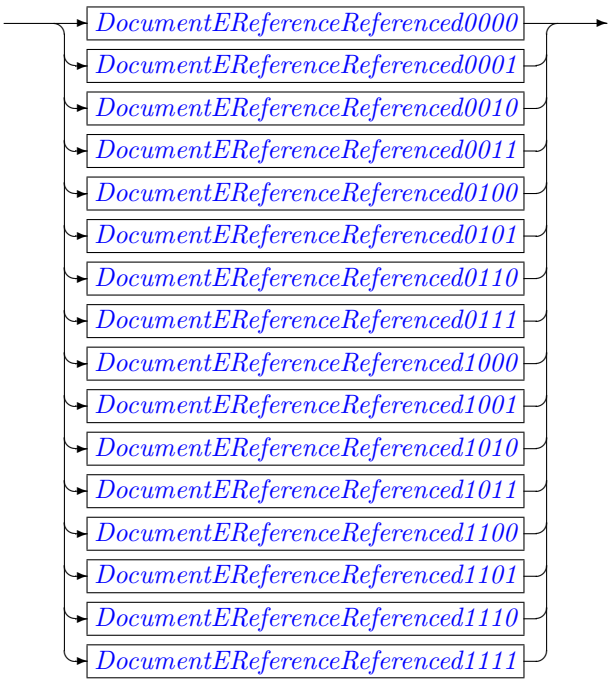
```
      </SUB-NODES>
    </CHILD>
  </CHILDREN>
</NODE>
```

8.4 Document Mappings for non-containment EReferences

Preconditions Document mapping is applicable if

$$\begin{aligned} & false == x.isTransient() \\ \wedge\ & false == x.isContainment() \\ \wedge\ & ExtendedMetaData.ELEMENT\_FEATURE == extendedMetaData.getFeatureKind(x) \end{aligned}$$

DocumentEReferenceReferenced



Note 1

	XMLPersistenceExtendeMetaData			
Mapping Rule	feature WrapperElement	feature Element	classifier WrapperElement	classifier Element
...EReferenceReferenced0000	false	false	false	false
...EReferenceReferenced0001	false	false	false	true
...EReferenceReferenced0010	false	false	true	false
...EReferenceReferenced0011	false	false	true	true
...EReferenceReferenced0100	false	true	false	false
...EReferenceReferenced0101	false	true	false	true
...EReferenceReferenced0110	false	true	true	false
...EReferenceReferenced0111	false	true	true	true
...EReferenceReferenced1000	true	false	false	false
...EReferenceReferenced1001	true	false	false	true
...EReferenceReferenced1010	true	false	true	false
...EReferenceReferenced1011	true	false	true	true
...EReferenceReferenced1100	true	true	false	false
...EReferenceReferenced1101	true	true	false	true
...EReferenceReferenced1110	true	true	true	false
...EReferenceReferenced1111	true	true	true	true

The following default mapping rules apply:

- If XMLPersistenceExtendedMetaData is not defined, then rule DocumentEReferenceReferenced0100 applies (default EMF persistence behaviour).
- If XMLPersistenceExtendedMetaData is defined and one of the keys featureWrapperElement, featureElement, classifierWrapperElement, classifierWrapper is missing, then the rule DocumentEReferenceReferenced1100 applies for EReferences with $true == x.isMany()$ and rule DocumentEReferenceReferenced0100 otherwise. Since these missing keys identify an error in the configuration, the (de)serializer shall log an error message.

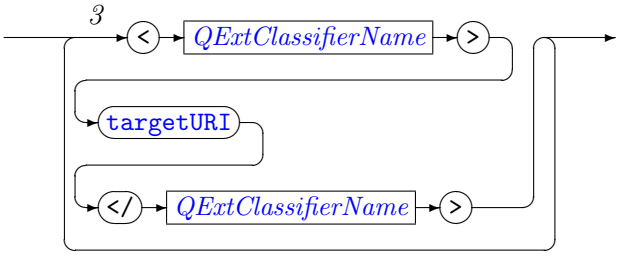
8.4.1 Document Mapping for EReferenceReferenced0000

not applicable

8.4.2 Document Mapping for EReferenceReferenced0001

EBNF - Syntax Diagram

DocumentEReferenceReferenced0001



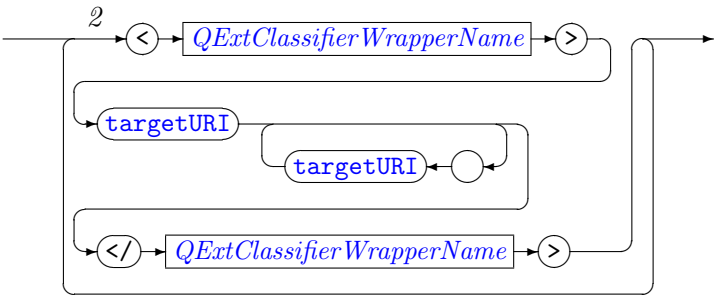
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REF>
        <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
        <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
        <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
      </RELATED-REF>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.3 Document Mapping for EReferenceReferenced0010

EBNF - Syntax Diagram

DocumentEReferenceReferenced0010



Document Example

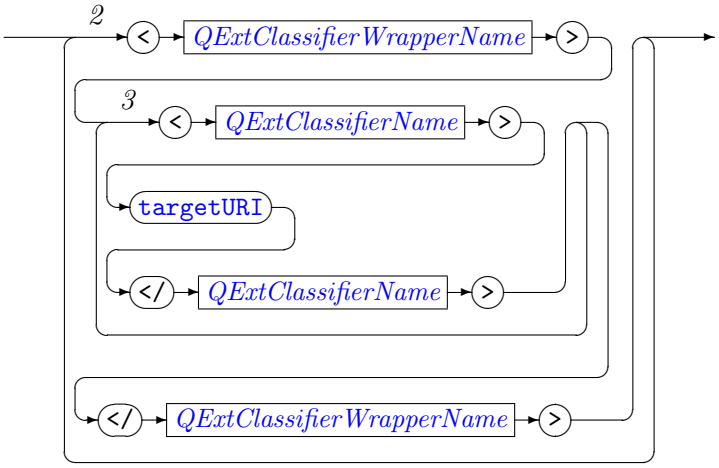
```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
```

```
    <NODE name="sourceNode">
      <NODE-REFTYPES>targetNode1 targetNode2</NODE-REFTYPES>
      <SUB-NODE-REFTYPES>targetSubNode</SUB-NODE-REFTYPES>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.4 Document Mapping for EReferenceReferenced0011

EBNF - Syntax Diagram

DocumentEReferenceReferenced0011



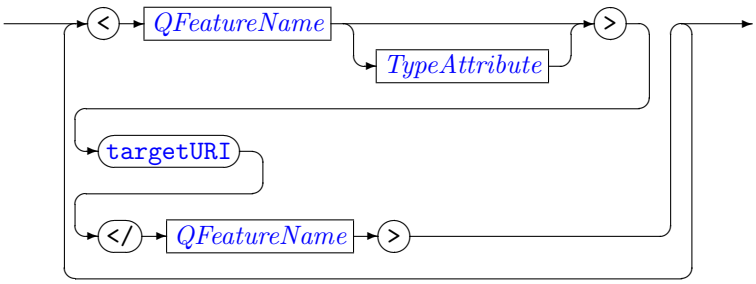
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <NODE-REFTYPES>
        <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
        <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
      </NODE-REFTYPES>
      <SUB-NODE-REFTYPES>
        <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
      </SUB-NODE-REFTYPES>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.5 Document Mapping for EReferenceReferenced0100

EBNF - Syntax Diagram

DocumentEReferenceReferenced0100



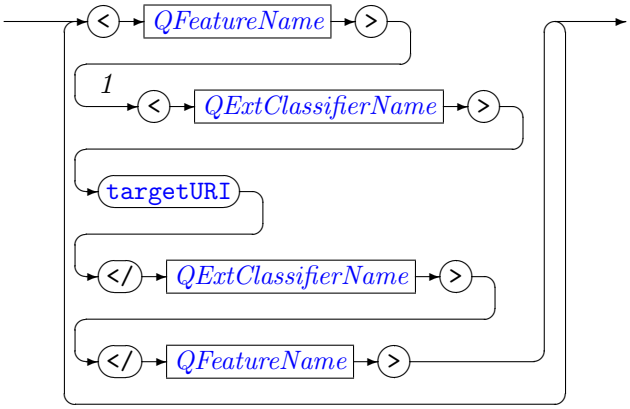
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REF>targetNode1</RELATED-REF>
      <RELATED-REF>targetNode2</RELATED-REF>
      <RELATED-REF TYPE="SUB-NODE">targetSubNode</RELATED-REF>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.6 Document Mapping for EReferenceReferenced0101

EBNF - Syntax Diagram

DocumentEReferenceReferenced0101



Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REF>
        <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
      </RELATED-REF>
      <RELATED-REF>
        <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
      </RELATED-REF>
      <RELATED-REF>
        <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
      </RELATED-REF>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.7 Document Mapping for EReferenceReferenced0110

[To do: todo]

Document Example

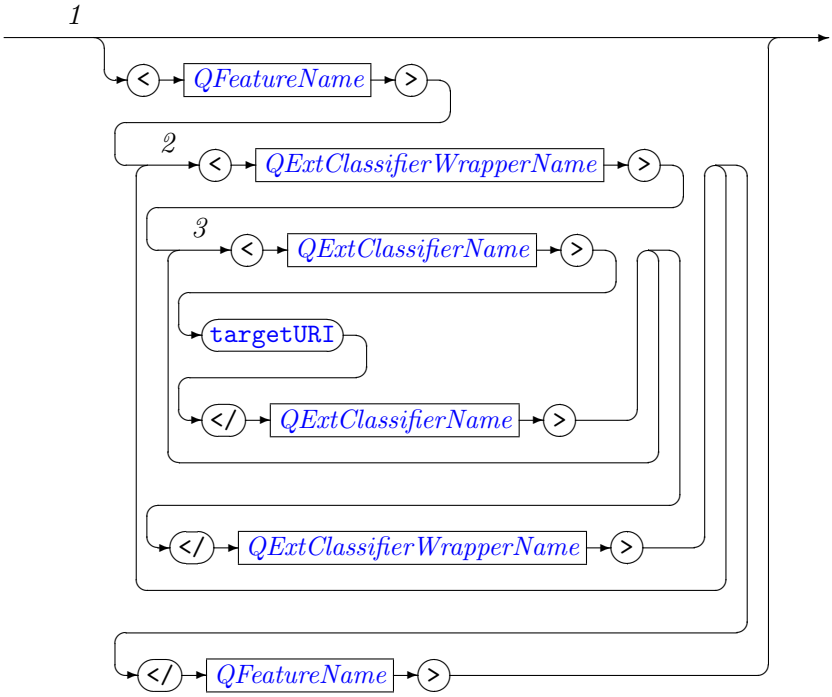
```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REF>
```

```
        <NODE-REFTYPES>targetNode1 targetNode2</NODE-REFTYPES>
        <SUB-NODE-REFTYPES>targetSubNode</SUB-NODE-REFTYPES>
      </RELATED-REF>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.8 Document Mapping for EReferenceReferenced0111

EBNF - Syntax Diagram

DocumentEReferenceReferenced0111



Document Example

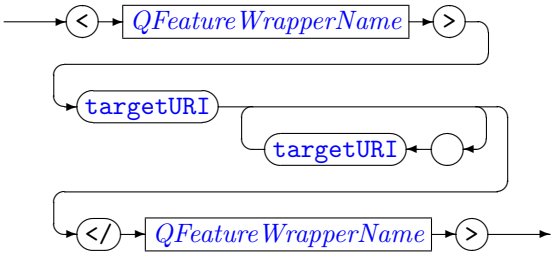
```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REF>
        <NODE-REFTYPES>
          <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
          <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
        </NODE-REFTYPES>
```

```
        <SUB-NODE-REFTYPES>
          <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
        </SUB-NODE-REFTYPES>
      </RELATED-REF>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.9 Document Mapping for EReferenceReferenced1000

EBNF - Syntax Diagram

DocumentEReferenceReferenced1000



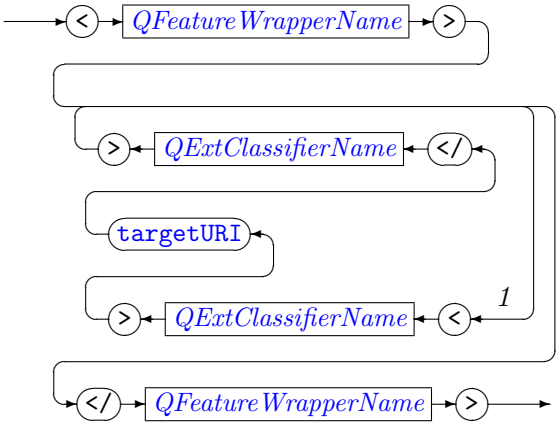
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>targetNodee targetNode2 SUB-NODE#targetSubNode</RELATED-REFS>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.10 Document Mapping for EReferenceReferenced1001

EBNF - Syntax Diagram

DocumentEReferenceReferenced1001



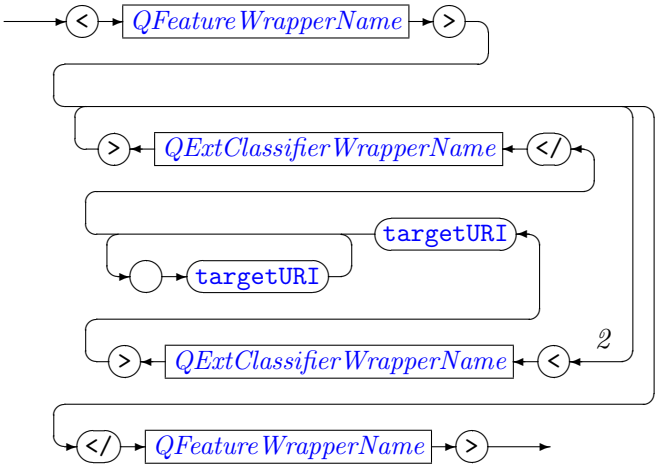
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>
        <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
        <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
        <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.11 Document Mapping for EReferenceReferenced1010

EBNF - Syntax Diagram

DocumentEReferenceReferenced1010



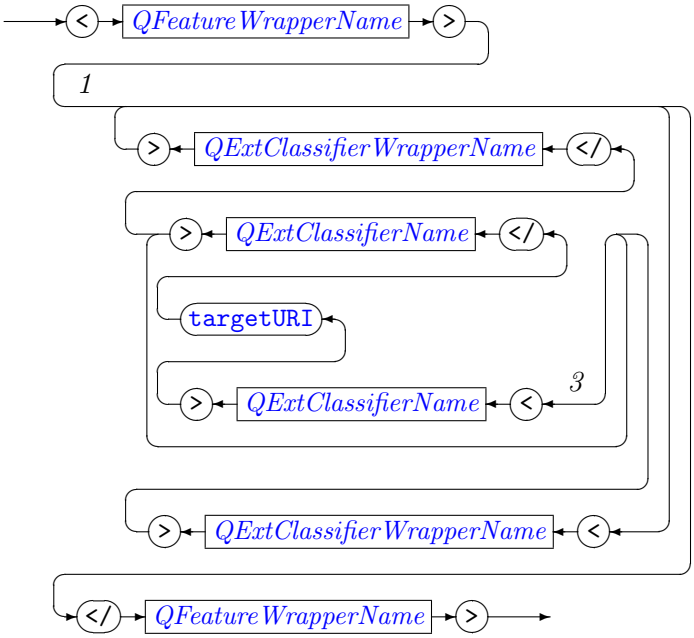
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>
        <NODE-REFTYPES>targetNode1 targetNode2</NODE-REFTYPES>
        <SUB-NODE-REFTYPES>targetSubNode</SUB-NODE-REFTYPES>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.12 Document Mapping for EReferenceReferenced1011

EBNF - Syntax Diagram

DocumentEReferenceReferenced1011



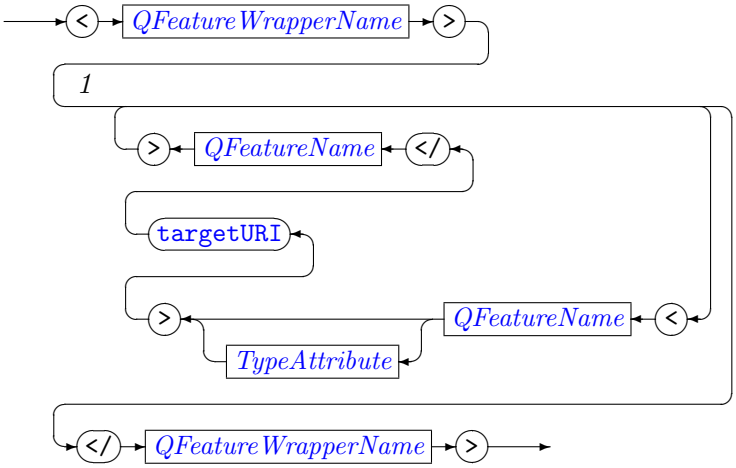
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
    <RELATED-REFS>
      <RELATED-REF>
        <NODE-REFTYPES>
          <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
          <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
        </NODE-REFTYPES>
        <SUB-NODE-REFTYPES>
          <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
        </SUB-NODE-REFTYPES>
      </RELATED-REF>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.13 Document Mapping for EReferenceReferenced1100

EBNF - Syntax Diagram

DocumentEReferenceReferenced1100



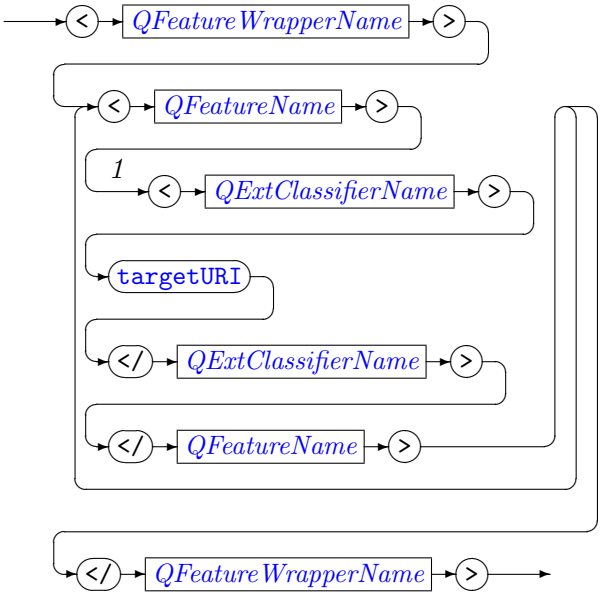
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>
        <RELATED-REF>targetNode1</RELATED-REF>
        <RELATED-REF>targetNode2</RELATED-REF>
        <RELATED-REF TYPE="SUB-NODE">targetSubNode</RELATED-REF>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.14 Document Mapping for EReferenceReferenced1101

EBNF - Syntax Diagram

DocumentEReferenceReferenced1101



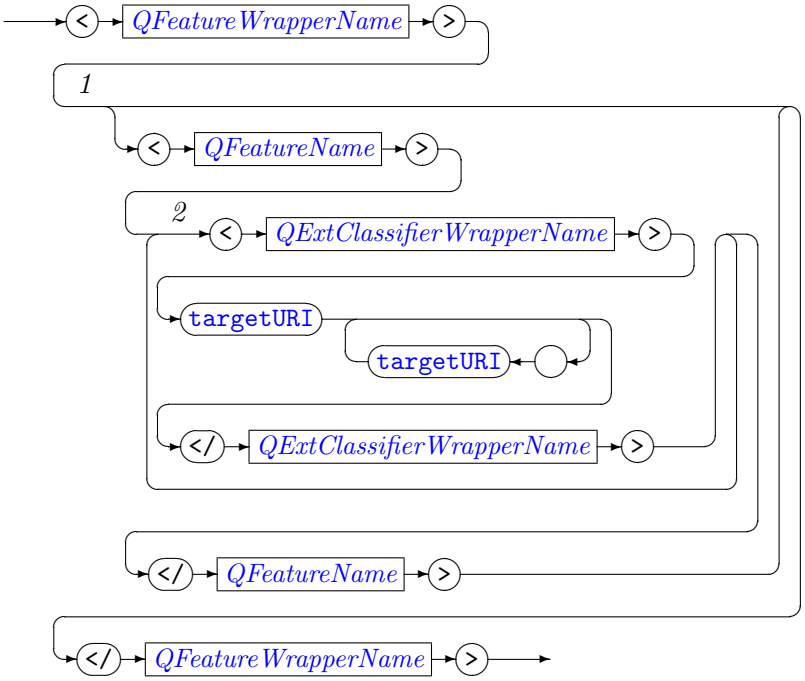
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>
        <RELATED-REF>
          <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
        </RELATED-REF>
        <RELATED-REF>
          <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
        </RELATED-REF>
        <RELATED-REF>
          <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
        </RELATED-REF>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1"/>
    <NODE name="targetNode2"/>
    <SUB-NODE name="targetSubNode"/>
  </CHILDREN>
</NODE>
```

8.4.15 Document Mapping for EReferenceReferenced1110

EBNF - Syntax Diagram

DocumentEReferenceReferenced1110



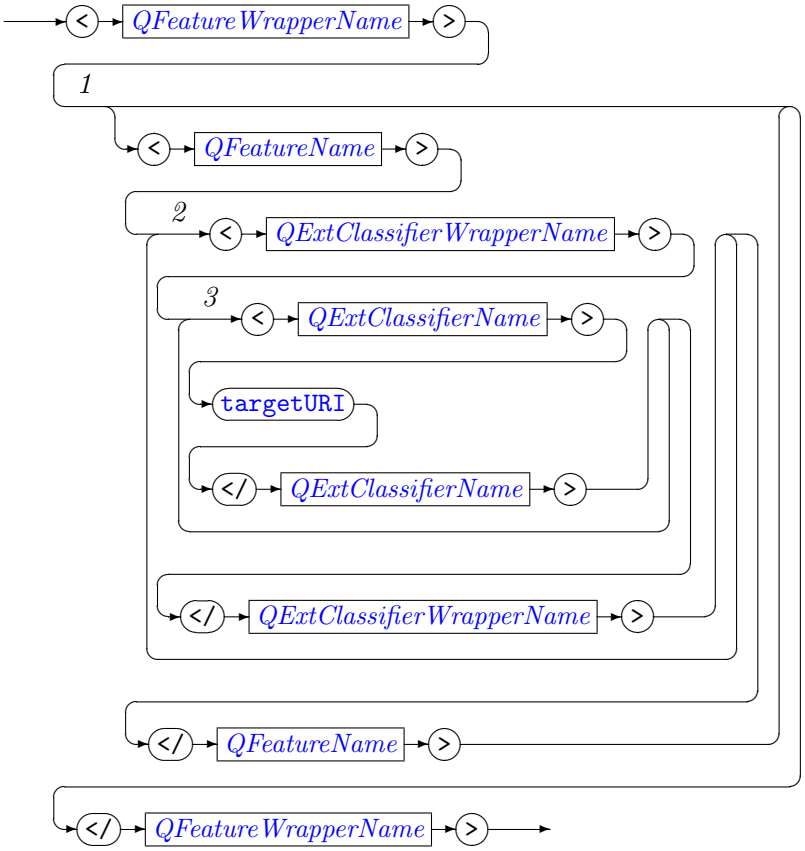
Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
    <RELATED-REFS>
      <RELATED-REF>
        <NODE-REFTYPES>targetNode1 targetNode2</NODE-REFTYPES>
        <SUB-NODE-REFTYPES>targetSubNode</SUB-NODE-REFTYPES>
      </RELATED-REF>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
  </CHILDREN>
</NODE>
```

8.4.16 Document Mapping for EReferenceReferenced1111

EBNF - Syntax Diagram

DocumentEReferenceReferenced1111



Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<NODE xmlns="NodesURI" name="root">
  <CHILDREN>
    <NODE name="sourceNode">
      <RELATED-REFS>
        <RELATED-REF>
          <NODE-REFTYPES>
            <NODE-REFTYPE>targetNode1</NODE-REFTYPE>
            <NODE-REFTYPE>targetNode2</NODE-REFTYPE>
          </NODE-REFTYPES>
          <SUB-NODE-REFTYPES>
            <SUB-NODE-REFTYPE>targetSubNode</SUB-NODE-REFTYPE>
          </SUB-NODE-REFTYPES>
        </RELATED-REF>
      </RELATED-REFS>
    </NODE>
    <NODE name="targetNode1" />
    <NODE name="targetNode2" />
    <SUB-NODE name="targetSubNode" />
```

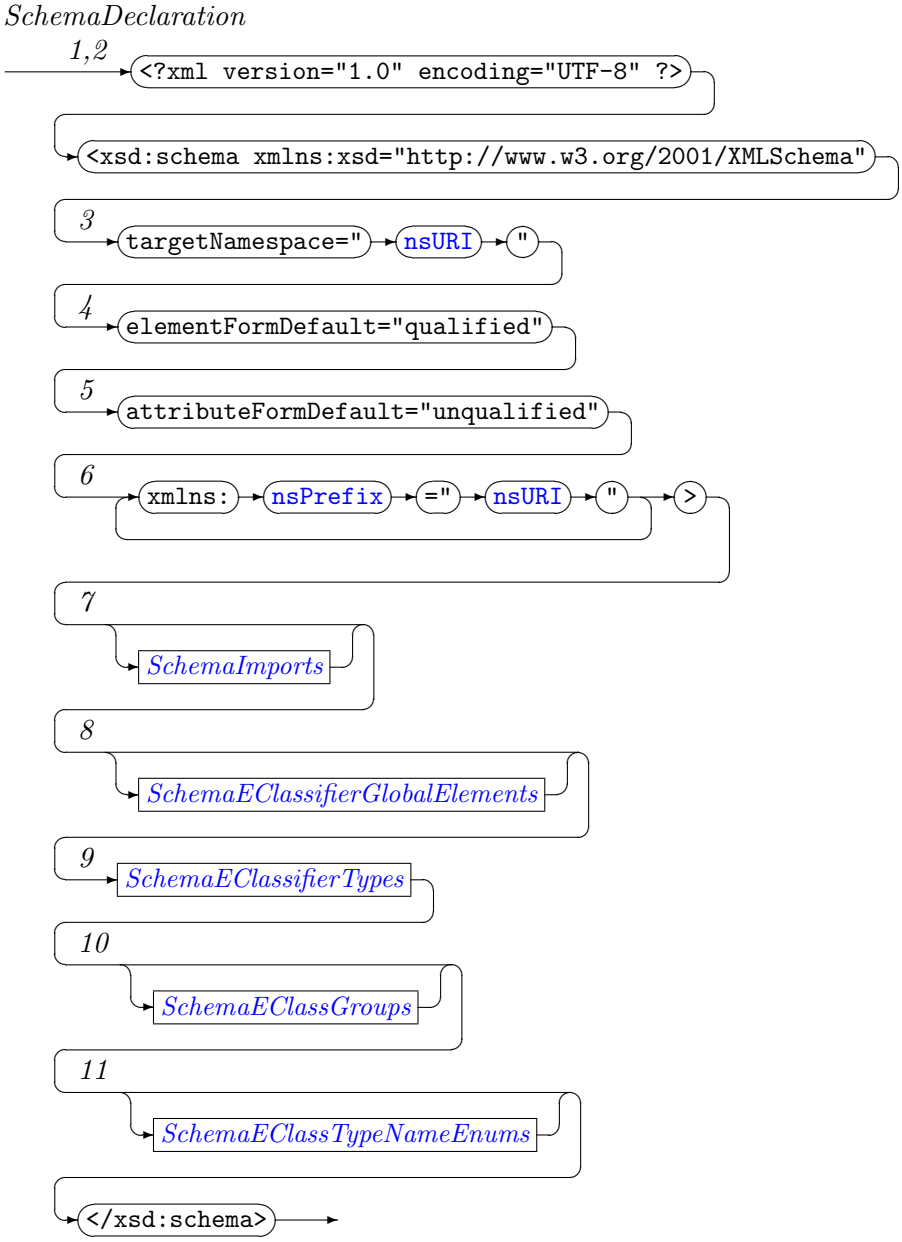
```
  </CHILDREN>
</NODE>
```

9 XML Schema Production

This chapter contains information for XML Schema developers.

9.1 XML Schema Declaration

EBNF - Syntax Diagram



Notes

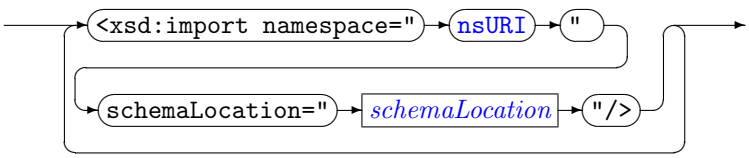
1. The version relates to this schema only and doesn't put any constraints on the XML version of the XML instance. Version 1.0 is sufficient since features that are only supported by XML1.1 are not required. Additionally, the XML1.1 specfication states: "XML 1.1 processors must be able to process both XML 1.0 and XML 1.1 documents. Programs which generate XML should generate XML 1.0, unless one of the specific features of XML 1.1 is required."[7]

2. The encoding relates to this schema only and dooesn't put any constraonts on the encoding of the XML instance. Encoding UTF-8 is selected since the XML 1.0 specification requires that "all XML processors must accept the UTF-8 and UTF-16 encodings of Unicode"[6] and UTF-8 requires less memory for the characters used in the XML Schema.
3. The targetNamespace is defined by the .nsURI attribute of the ePackage.
4. The elementFormDefault is set to "qualified" in order to enforce, that all elements in the XML instance are properly qualified by a namespace prefix. This simplifies implementation of simple XML scripts since they do not need to implement complex calculations for namespace detection.
5. The attributeFormDefault is set to "unqualified" in order to avoid namespace prefixes for each attribute. This reduces XML file size.
6. For current ePackage and all referenced ePackages a namespace prefix to namespace mapping is required. The nsPrefix and nsURI are related to the corresponding attributes of EPackage.
7. If the metamodel references or contains EClassifiers or EStructuralFeatures that define a namespace which is different from the namespace of this EPackage, then import statements are required.
8. If the metamodel contains EClasses that are annotated with 'xmlGlobalElement=true' then global XML element declarations are required. Note: A schema that do not define any global elements are valid and can be imported by other XML schema. However, they cannot be used standalone for validation, since no root XML element is defined.
9. For all EClassifiers that are not abstract XML type declarations are required.
10. For reduction of XML Schema size, the Schema production can be configured leverage reusable named xsd:groups (key=useElementGroups, key=useAttributeGroups).
11. For all referenced EClasses that have subtypes enumerations are produced that list the valid type and its subtypes.

9.2 XML Schema Imports

EBNF - Syntax Diagram

SchemaImports



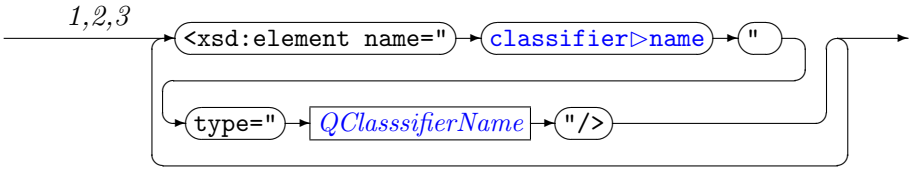
Notes

1. For all referenced ePackages an import statement is required. The nsURI is related to the nsURI of the referenced EPackage. The schemaLocation is derived from its meta data (key: schemaLocation).
2. Additionally, imports are required for all externally referenced namespaces (e.g. "http://www.w3.org/XML/1998/namespace"). These namespaces are identfied by the EPackage annotation "externalSchemaLoccations".
3. No import is required for references to simple Ecore data types such as EString, EFLoat, etc. Those datatypes are mapped to built in xsd types.
4. The list of imports shall be ordered alphabetically by the nsURI

9.3 XML Schema Global Elements

EBNF - Syntax Diagram

SchemaEClassifierGlobalElements



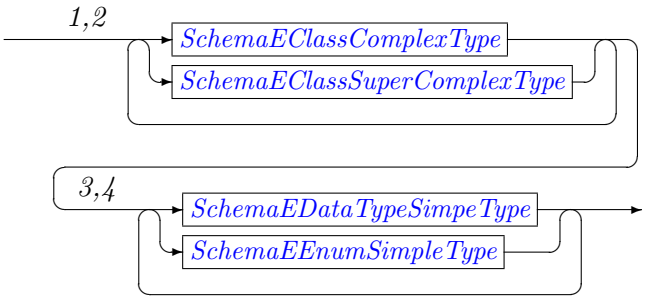
Notes

1. For all EDataTypes that are annotated with 'xmlGlobalElement=true' a global element declaration is required.
2. For all EClasses that are annotated with 'xmlGlobalElement=true' and are not abstract a global element declaration is required.
3. The list of global elements shall be ordered alphabetically by the XML classifierName.

9.4 XML Schema Types

EBNF - Syntax Diagram

SchemaEClassifierTypes



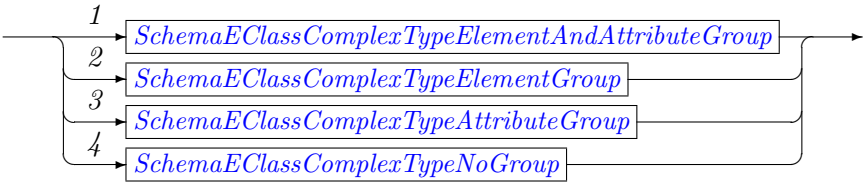
Notes

1. Produce complex type for all EClasses that are not abstract.
2. The list of simple types shall be ordered alphabetically by the XML classifierName
3. Produce simple type for all EDataTypes that are not predefined ecore datatypes such as EString, EFloat, etc.
4. The list of complex types shall be ordered alphabetically by the XML classifierName

9.4.1 XML Schema Complex Types

EBNF - Syntax Diagram

SchemaEClassComplexType



Notes

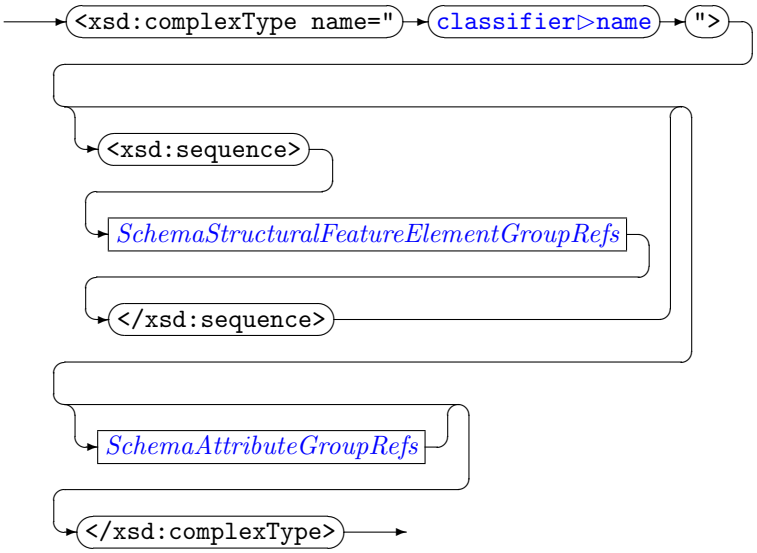
1. use SchemaEClassComplexTypeElementAndAttributeGroup if the containing EPackage is annotated with useElementGroup=true and useAttributeGroup=true.
2. use SchemaEClassComplexTypeElementGroup if the containing EPackage is annotated with useElementGroup=true and useAttributeGroup=false.
3. use SchemaEClassComplexTypeAttributeGroup if the containing EPackage is annotated with useElementGroup=false and useAttributeGroup=true.

4. use SchemaEClassComplexTypeNoGroup if the containing EPackage is annotated with useElementGroup=false and useAttributeGroup=false.

9.4.2 XML Schema Complex Types with Element and Attribute Groups

EBNF - Syntax Diagram

SchemaEClassComplexTypeElementAndAttributeGroup



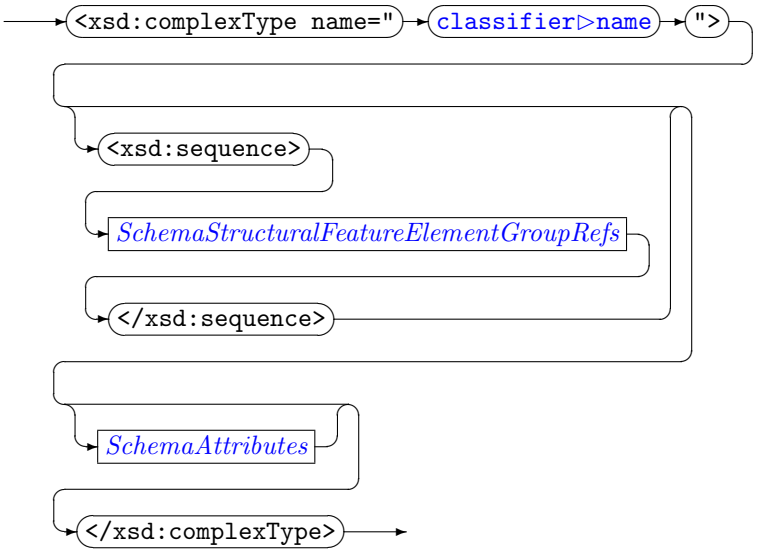
Notes

1. xsd:all may contain XML element declarations and annotations only [8]. Thus, xsd:all is not used here.

9.4.3 XML Schema Complex Types with Element Groups

EBNF - Syntax Diagram

SchemaEClassComplexTypeElementGroup



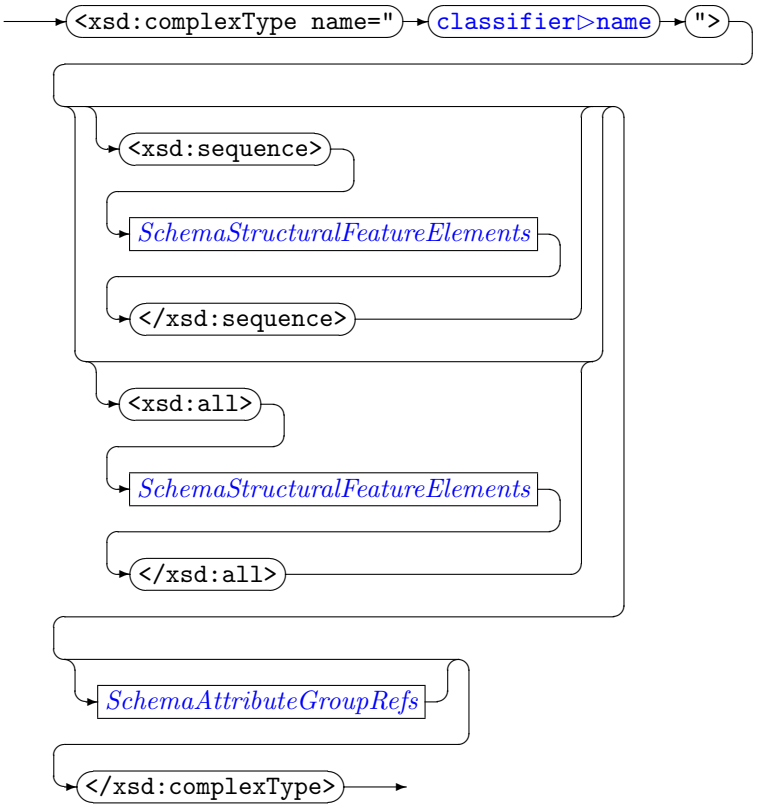
Notes

1. xsd:all may contain XML element declarations and annotations only [8]. Thus, xsd:all is not used here.

9.4.4 XML Schema Complex Types with Attribute Groups

EBNF - Syntax Diagram

SchemaEClassComplexTypeAttributeGroup



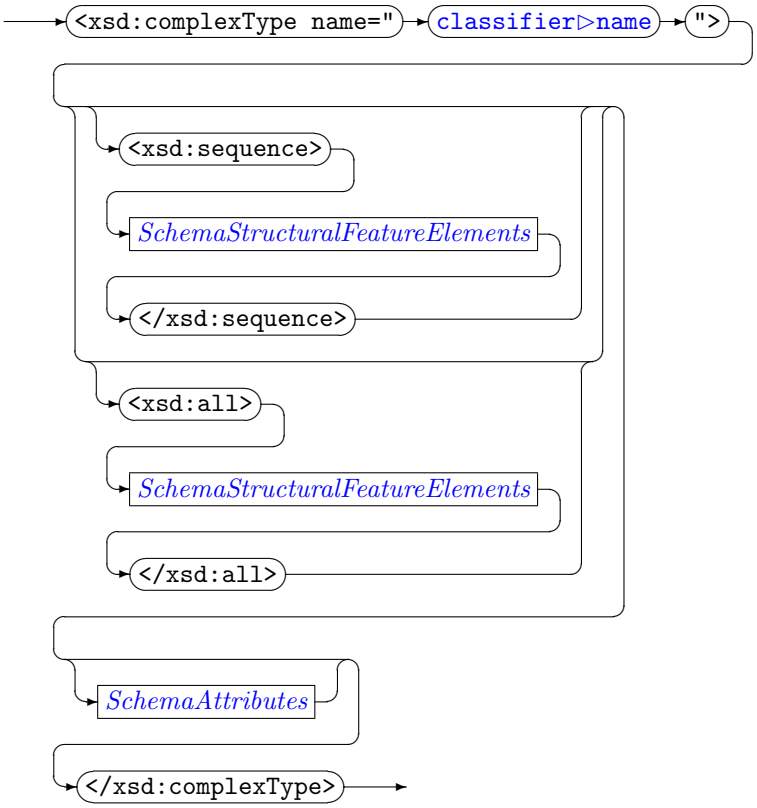
Notes

1.

9.4.5 XML Schema Complex Types without Groups

EBNF - Syntax Diagram

SchemaEClassComplexTypeElementNoGroup



Notes

1.

9.5 XML Schema Simple Types

9.5.1 Ecore Types

For all mappings of EStructural features the eType shall be mapped as defined in the following table

Model type	Schema type
ecore:EBoolean	xsd:boolean
ecore:EBooleanObject	xsd:boolean
ecore:EDouble	xsd:double
ecore:EDoubleObject	xsd:double
ecore:EFloat	xsd:float
ecore:EFloatObject	xsd:float
ecore:EInt	xsd:int
ecore:EIntegerObject	xsd:int
ecore:ELong	xsd:long
ecore:ELongObject	xsd:long
ecore:EString	xsd:string
custom EDataType	<pkg.prefix>: classifierName

9.5.2 Custom Simple Types

Custom EDataTypes are mapped to custom simpleType which restrict from the following schema types:

Java instanceClass	Schema base Type
java.lang.boolean	xsd:boolean
java.lang.Boolean	xsd:boolean
java.lang.char	xsd:string
java.lang.Char	xsd:string
java.lang.double	xsd:double
java.lang.Double	xsd:double
java.lang.byte	xsd:byte
java.lang.Byte	xsd:byte
java.lang.float	xsd:float
java.lang.Float	xsd:float
java.lang.int	xsd:int
java.lang.Integer	xsd:int
java.lang.long	xsd:long
java.lang.Long	xsd:long
java.lang.short	xsd:short
java.lang.Short	xsd:short
java.lang.String	xsd:string
java.math.BigInteger	xsd:integer

EBNF - Syntax Diagram

SchemaEDataTypeSimpeType

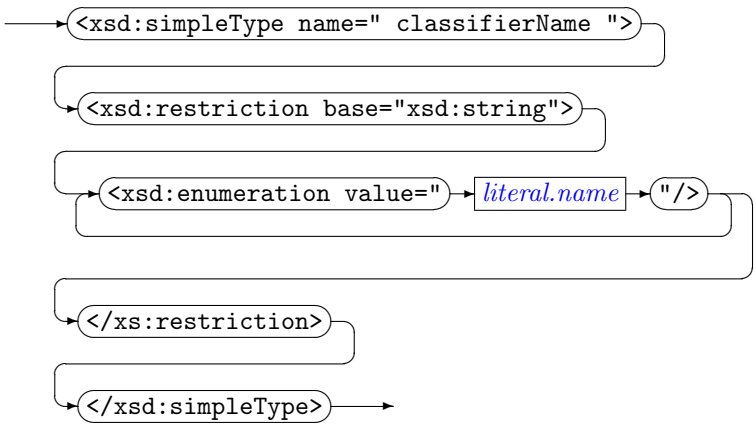
Notes

1. The properties of the XML simple type restriction are derived from annotations of the EDataType.
2. These annotations are used for definition of restrictions in XML schema as well as for validators that are generated by the EMF code generator.
3. The valid properties of each XML schema type are defined in the XML Schema Datatypes Specification[9]

9.5.3 Enumerations

EBNF - Syntax Diagram

SchemaEEnumSimpleType



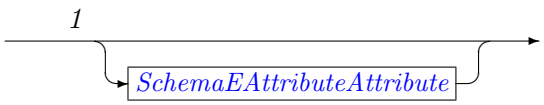
Notes

1.

9.6 XML Schema Attribute Mappings

EBNF - Syntax Diagram

SchemaAttributes



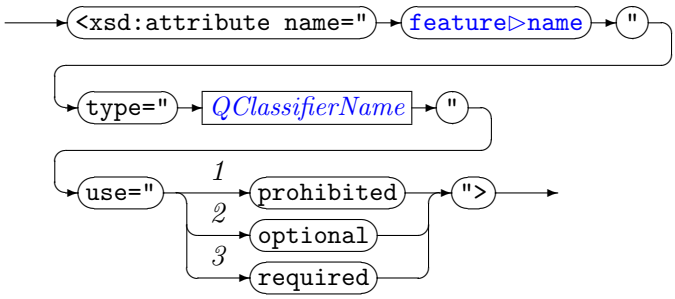
Notes

1. create attribute definition for all EAttributes with kind=attribute

9.6.1 XML Schema Attribute Mapping EAttributeAttribute

EBNF - Syntax Diagram

SchemaEAttributeAttribute



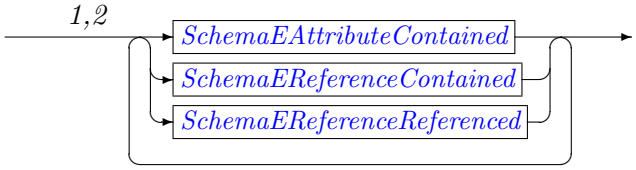
Notes

1. probibited if upperBound=0.
2. optional if lowerBound=0 and upperBound>0
3. required if lowerBound>0 and upperBound>0

9.7 XML Schema Element Mappings

EBNF - Syntax Diagram

SchemaStructuralFeatureElements



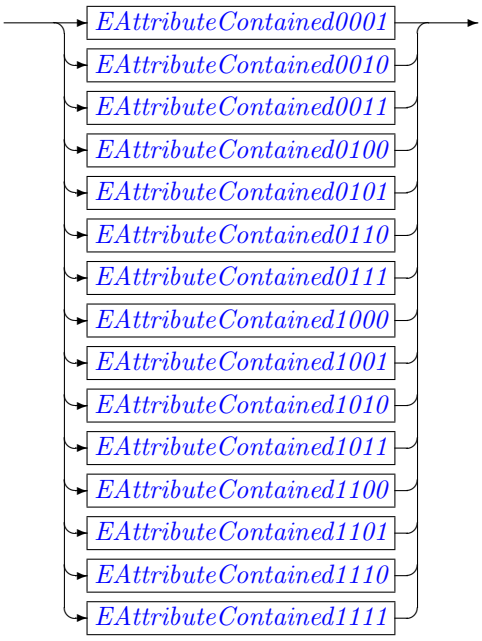
Notes

1. Create SchemaStructuralFeature for all EStructuralFeatures which are annotated with "kind=element" of the EClass including its super types .
2. The order of the elements is defined by the Ecore getEAllStructuralFeatures() call. (depth first navigation of eSuperType reference, order of super types as given by the eSuperTypes-EList, order of EStructuralFeatures within the EClass as defined by the eStructuralFeatures-EList)

9.8 XML Schema Element Mapping for EAttributes

EBNF - Syntax Diagram

SchemaEAttributeContained



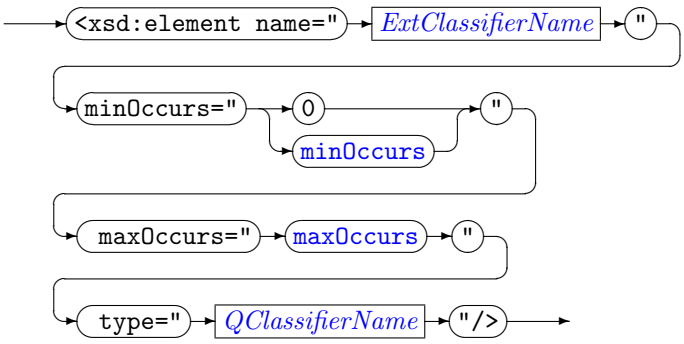
9.8.1 XML Schema Element Mapping EAttributeContained0000

not applicable

9.8.2 XML Schema Element Mapping EAttributeContained0001

EBNF - Syntax Diagram

EAttributeContained0001



Schema Example

```
<xsd:element name="STRING" minOccurs="0" maxOccurs="unbounded" type="node:STRING"/>
```

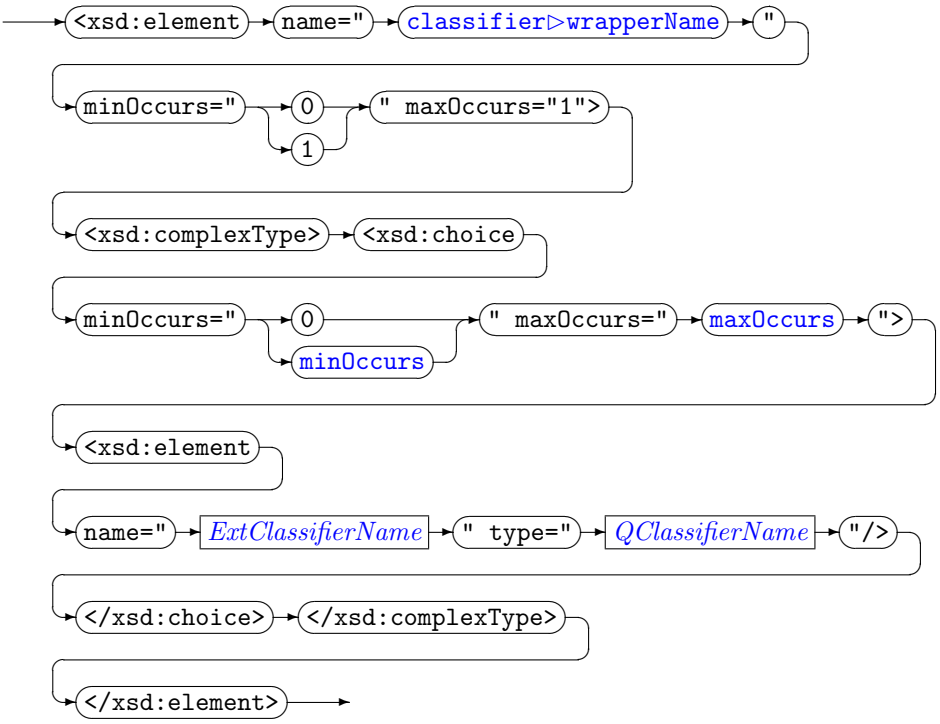
9.8.3 XML Schema Element Mapping EAttributeContained0010

[To do: todo]

9.8.4 XML Schema Element Mapping EAttributeContained0011

EBNF - Syntax Diagram

EAttributeContained0011



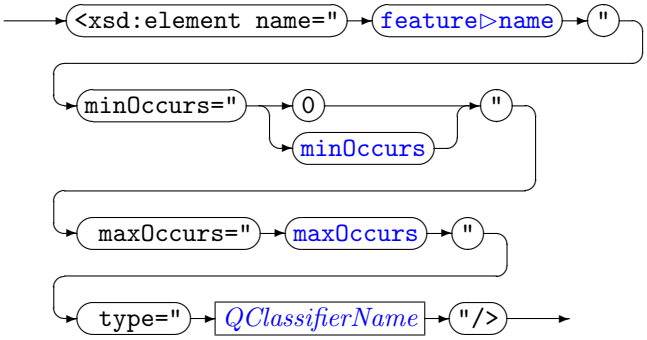
Schema Example

```
<xsd:element name="STRING-VALUES" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="STRING" type="node:STRING"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.8.5 XML Schema Element Mapping EAttributeContained0100

EBNF - Syntax Diagram

EAttributeContained0100



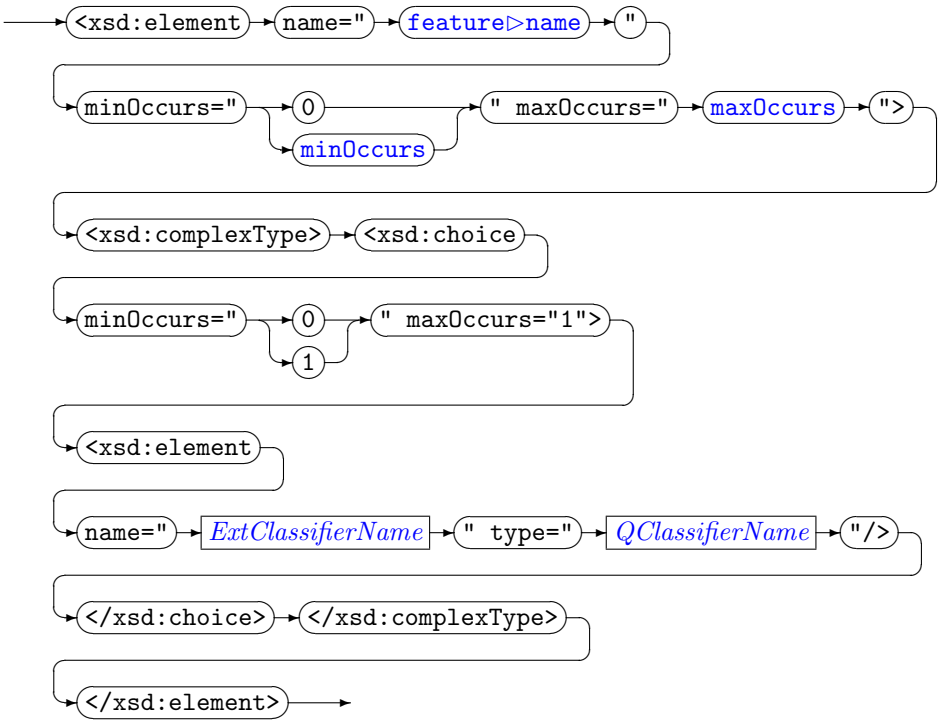
Schema Example

```
<xsd:element name="PROPERTY" minOccurs="0" maxOccurs="unbounded" type="node:STRING"/>
```

9.8.6 XML Schema Element Mapping EAttributeContained0101

EBNF - Syntax Diagram

EAttributeContained0101



Schema Example

```
<xsd:element name="PROPERTY" minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="1" >
      <xsd:element name="STRING" type="node:STRING"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.8.7 XML Schema Element Mapping EAttributeContained0110

[To do: todo]

9.8.8 XML Schema Element Mapping EAttributeContained0111

[To do: todo]

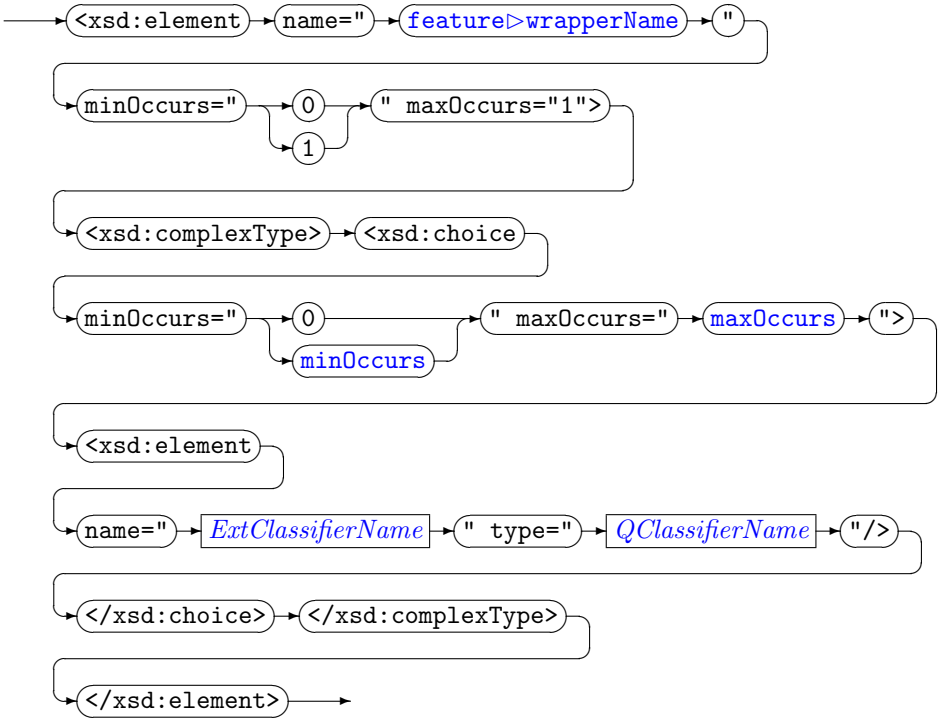
9.8.9 XML Schema Element Mapping EAttributeContained1000

[To do: todo]

9.8.10 XML Schema Element Mapping EAttributeContained1001

EBNF - Syntax Diagram

EAttributeContained1001



Schema Example

```
<xsd:element name="PROPERTIES" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="STRING" type="node:STRING"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.8.11 XML Schema Element Mapping EAttributeContained1010

[To do: todo]

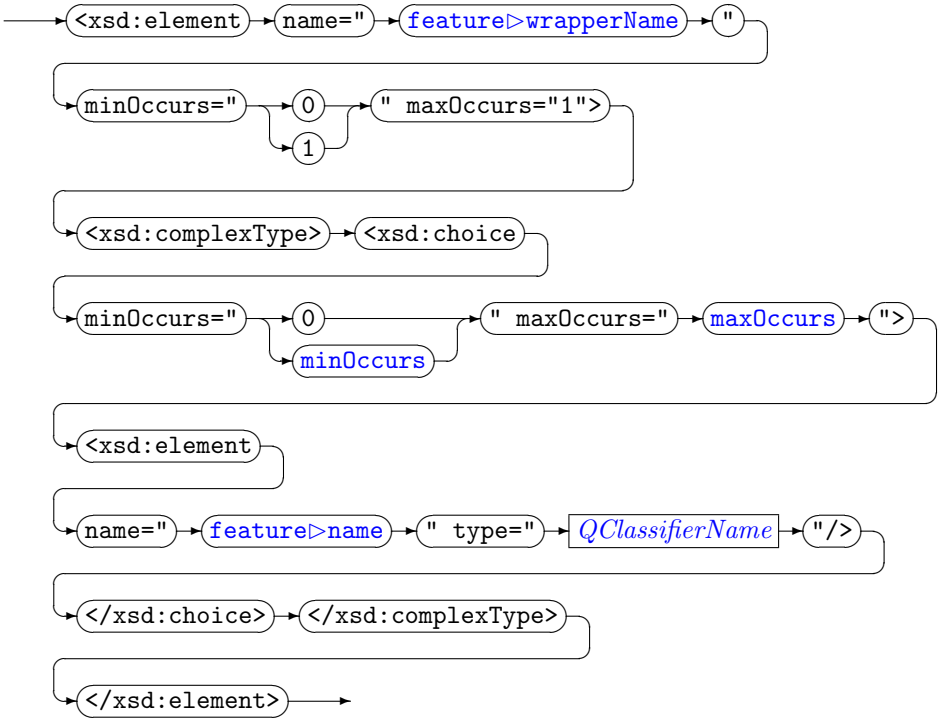
9.8.12 XML Schema Element Mapping EAttributeContained1011

[To do: todo]

9.8.13 XML Schema Element Mapping EAttributeContained1100

EBNF - Syntax Diagram

EAttributeContained1100



Schema Example

```
<xsd:element name="PROPERTIES" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="PROPERTY" type="node:STRING"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.8.14 XML Schema Element Mapping EAttributeContained1101

[To do: todo]

9.8.15 XML Schema Element Mapping EAttributeContained1110

[To do: todo]

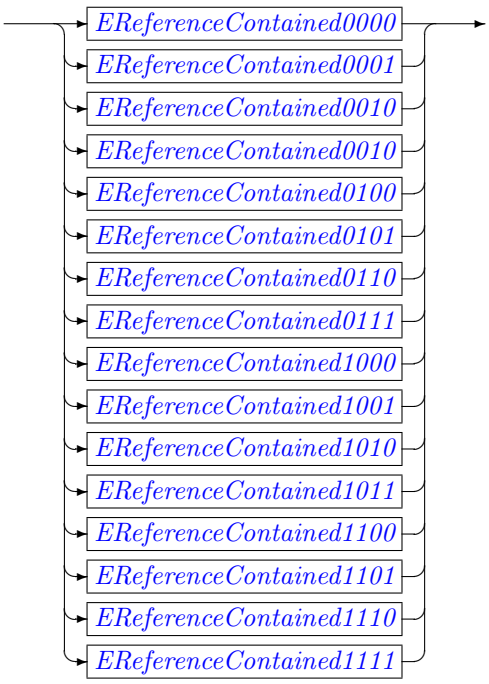
9.8.16 XML Schema Element Mapping EAttributeContained1111

[To do: todo]

9.9 XML Schema Mappings for containment EReferences

EBNF - Syntax Diagram

SchemaEReferenceContained



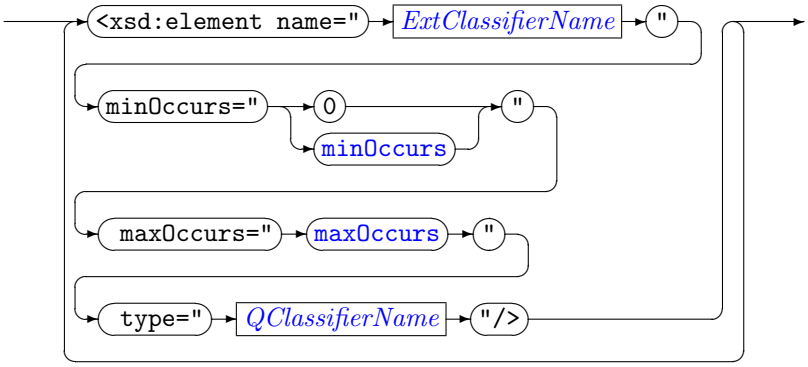
9.9.1 XML Schema Element Mapping EReferenceContained0000

[To do: todo]

9.9.2 XML Schema Element Mapping EReferenceContained0001

EBNF - Syntax Diagram

EReferenceContained0001



Notes

1. The list of elements is ordered alphabetically by the ExtClassifierName.

Schema Example

```
<xsd:element name="NODE" minOccurs="0" maxOccurs="unbounded" type="node:NODE"/>
<xsd:element name="SUB-NODE" minOccurs="0" maxOccurs="unbounded" type="node:SUB-NODE"/>
```

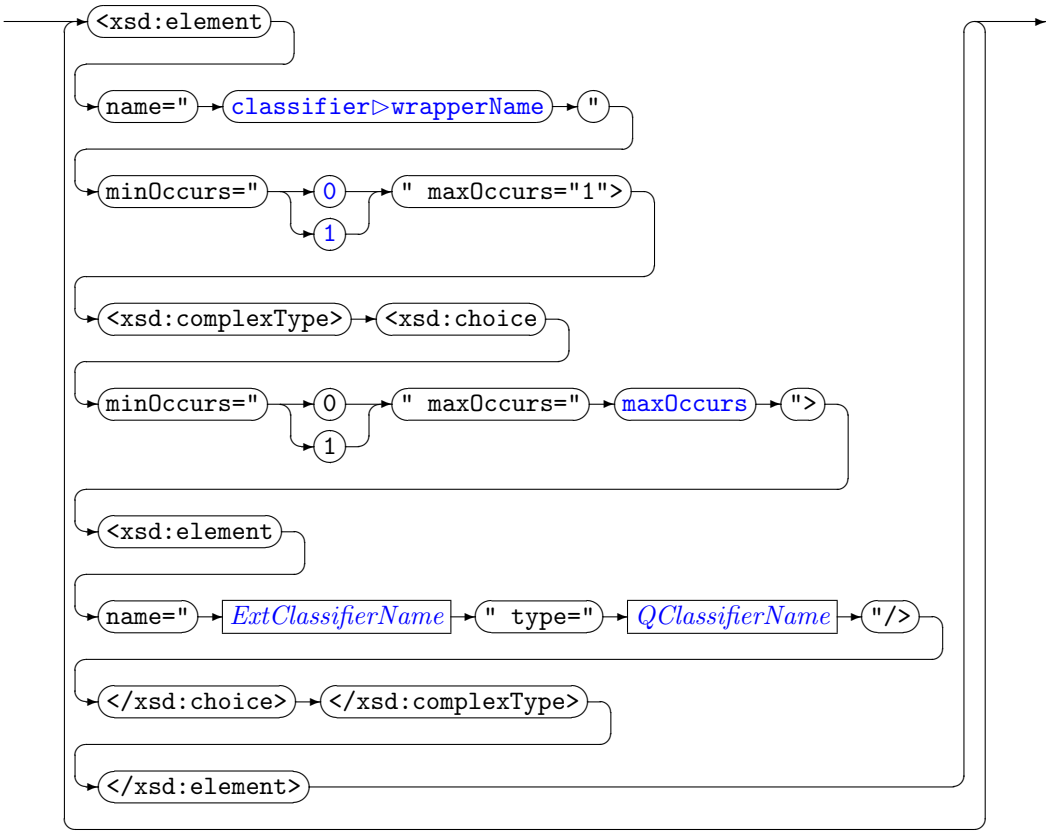
9.9.3 XML Schema Element Mapping EReferenceContained0010

[To do: todo]

9.9.4 XML Schema Element Mapping EReferenceContained0011

EBNF - Syntax Diagram

SchemaEReferenceContained0011



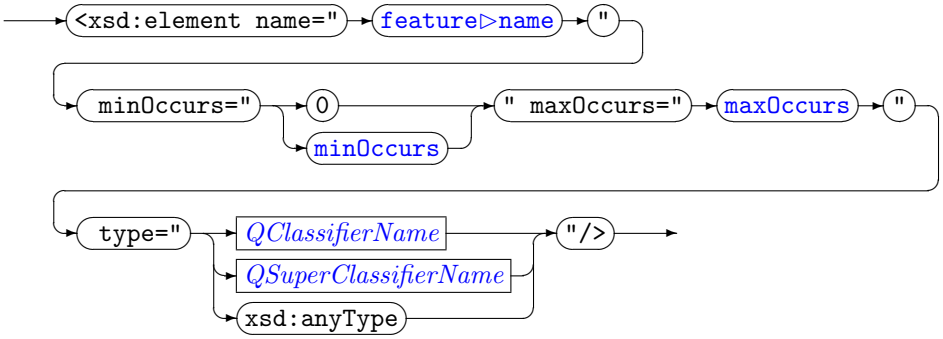
Schema Example

```
<xsd:element name="NODES" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="NODE" type="node:NODE"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="SUB-NODES" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="SUB-NODE" type="node:SUB-NODE"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.9.5 XML Schema Element Mapping EReferenceContained0100

EBNF - Syntax Diagram [To do: the EBNF is wrong FIXME]

SchemaEReferenceContained0100



- If the eType of the eReference has subclasses, then xsd:anyType shall be used instead of QClassifierName.

[To do: add concept of gathering all elements and attributes of all subtypes in order to build a superset that can be used for schema validation. This is not very strict but it is better than anyType.]

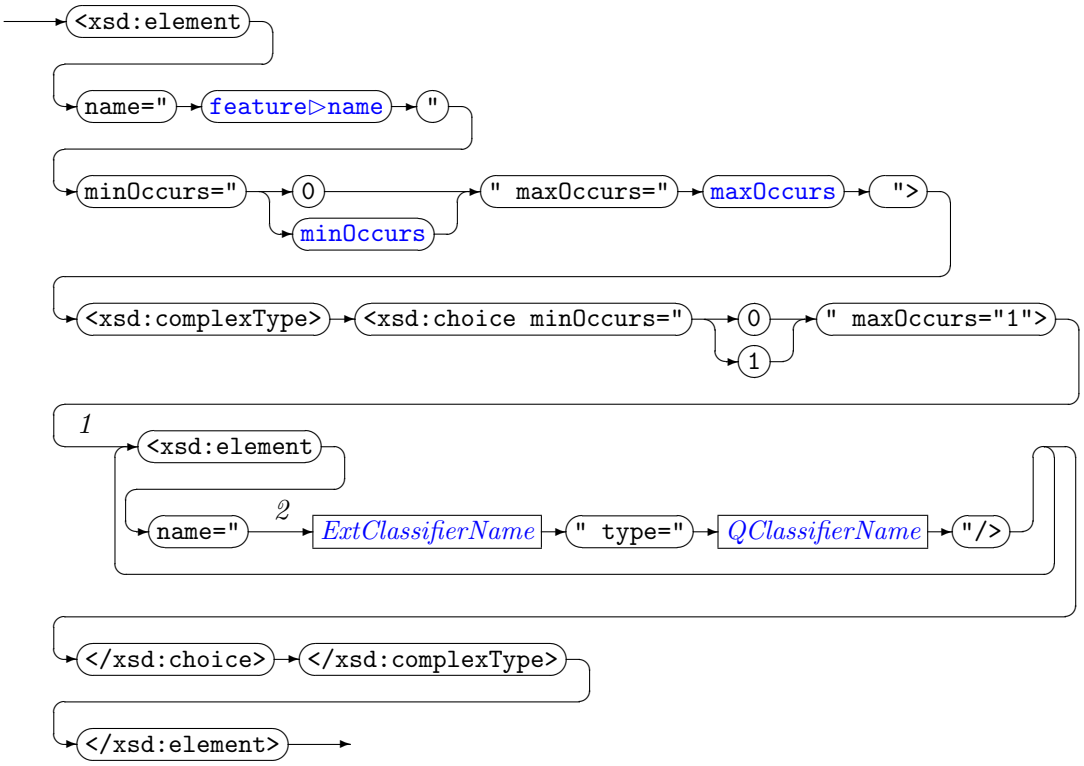
Schema Example

```
<xsd:element name="CHILD" minOccurs="0" maxOccurs="unbounded" type="node:NODE"/>

<xsd:element name="CHILD" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
           <xsd:group ref="node:NODE--SUPERTYPE-GROUP"/>
           <xsd:attribute name="TYPE" value="node:NODE--SUBTYPES-ENUM"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="CHILD" minOccurs="0" maxOccurs="unbounded" type="xsd:anyType"/>
```

9.9.6 XML Schema Element Mapping EReferenceContained0101

EBNF - Syntax Diagram

SchemaEReferenceContained0101



Notes

1. The complexType defines XML elements that represent allowed contained concrete types. An element is created for the following EClasses:
   (a) the eType of the EReference if it is not abstract
   (b) all direct and indirect subclasses of the eType if they are not abstract
2. The XML elements are ordered alphabetically by the ExtClassifierName.

Schema Example

```
<xsd:element name="CHILD" minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="1" >
      <xsd:element name="NODE" type="node:NODE"/>
      <xsd:element name="SUB-NODE" type="node:SUB-NODE"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.9.7 XML Schema Element Mapping EReferenceContained0110

[To do: todo]

9.9.8 XML Schema Element Mapping EReferenceContained0111

[To do: todo]

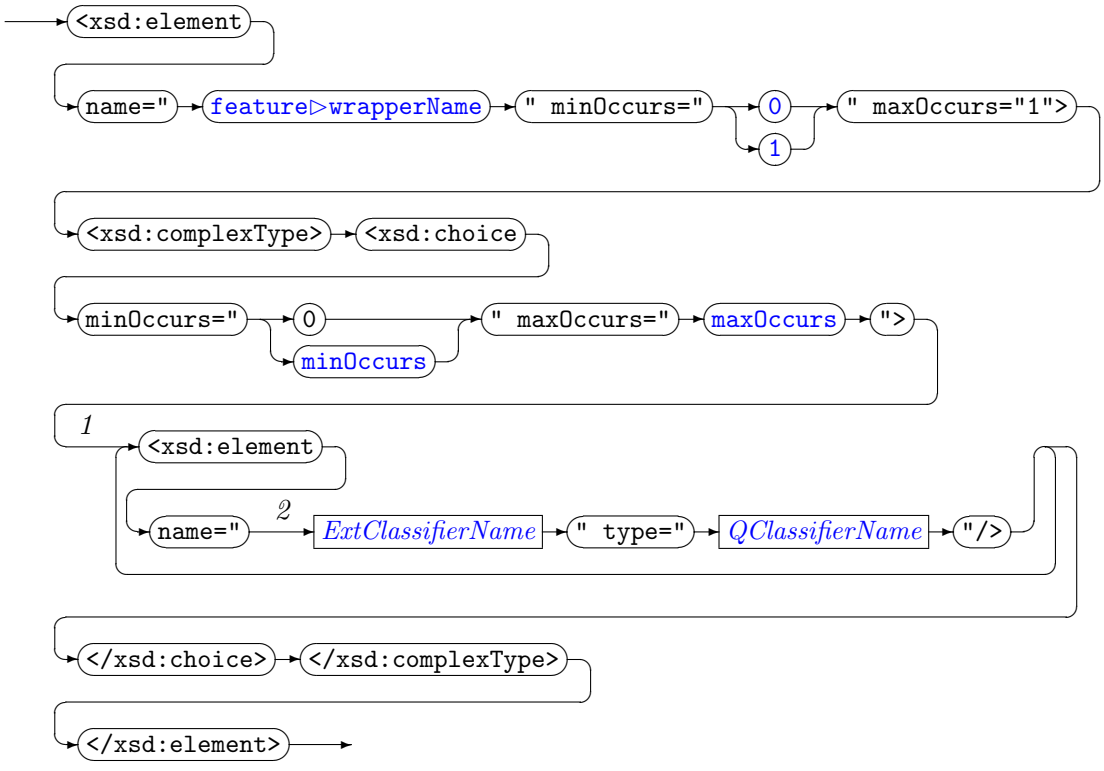
9.9.9 XML Schema Element Mapping EReferenceContained1000

[To do: todo]

9.9.10 XML Schema Element Mapping EReferenceContained1001

EBNF - Syntax Diagram

SchemaEReferenceContained1001



Notes

1. The complexType defines XML elements that represent allowed contained concrete types. An element is created for the following EClasses:
   (a) the eType of the EReference if it is not abstract
   (b) all direct and indirect subclasses of the eType if they are not abstract
2. The XML elements are ordered alphabetically by the ExtClassifierName.

Schema Example

```
<xsd:element name="CHILDREN" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="NODE" type="node:NODE"/>
      <xsd:element name="SUB-NODE" type="node:SUB-NODE"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.9.11 XML Schema Element Mapping EReferenceContained1010

[To do: todo]

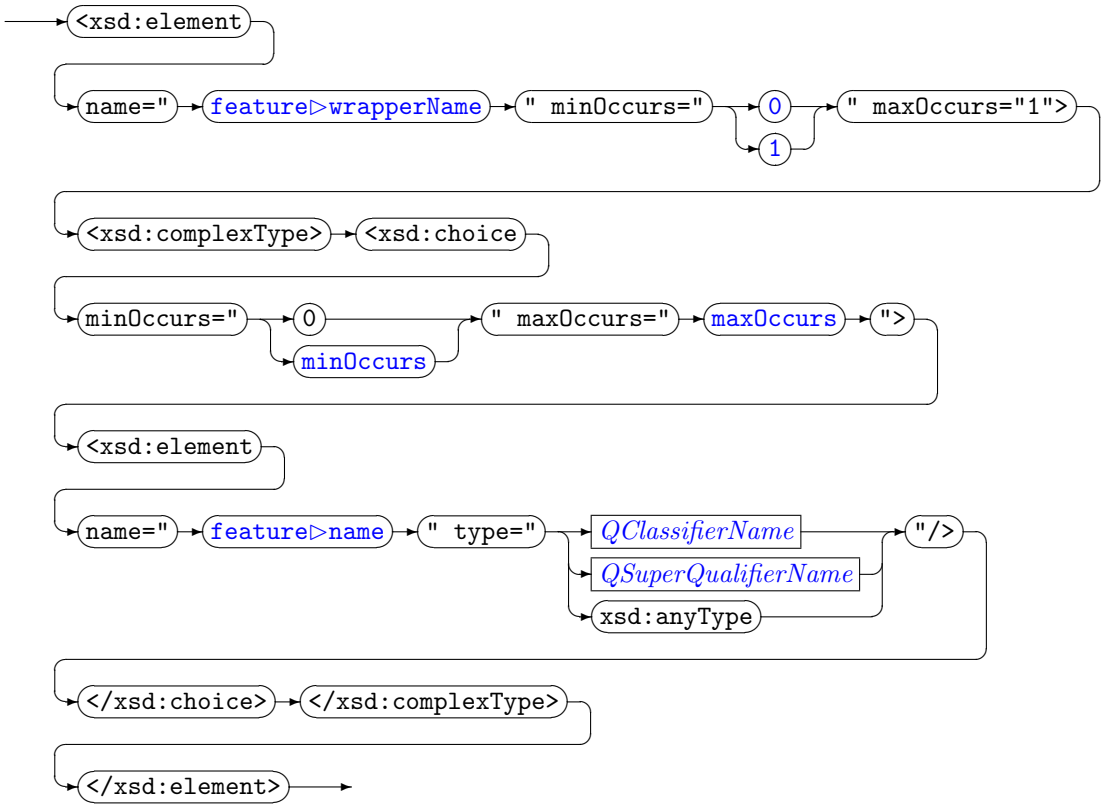
9.9.12 XML Schema Element Mapping EReferenceContained1011

[To do: todo]

9.9.13 XML Schema Element Mapping EReferenceContained1100

EBNF - Syntax Diagram [To do: the EBNF is wrong FIXME]

SchemaEReferenceContained1100



- If the eType of the eReference has subclasses, then xsd:anyType shall be used instead of QClassifierName.

[To do: todo add concept of gathering all elements and attributes of all subtypes in order to build a superset that can be used for schema validation. This is not very strict but it is better than anyType.]

Schema Example

```
<xsd:element name="CHILDREN" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="CHILD" type="node:NODE"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>

<xsd:element name="CHILDREN" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="CHILD">
        <xsd:complexType>
```

```
                <xsd:group ref="node:NODE--SUPERTYPE-GROUP"/>
                <xsd:attribute name="TYPE" value="node:NODE--SUBTYPES-ENUM"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>

<xsd:element name="CHILDREN" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="CHILD" type="xsd:anyType"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.9.14 XML Schema Element Mapping EReferenceContained1101

[To do: todo]

9.9.15 XML Schema Element Mapping EReferenceContained1110

[To do: todo]

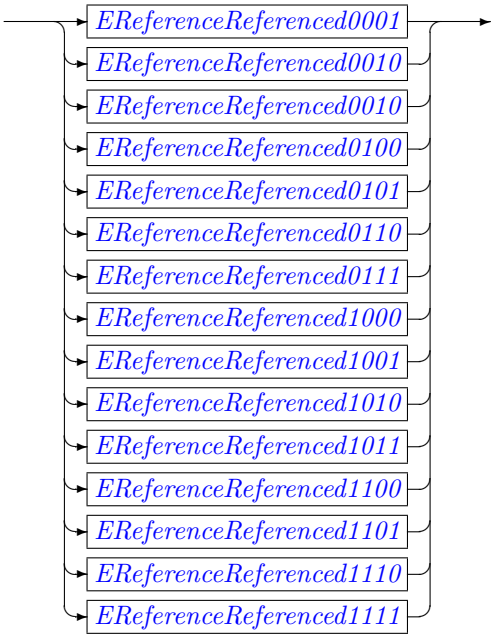
9.9.16 XML Schema Element Mapping EReferenceContained1111

[To do: todo]

9.10 XML Schema Mappings for Non-Containment EReference

EBNF - Syntax Diagram

SchemaEReferenceReferenced



9.10.1 XML Schema Element Mapping EReferenceReferenced0000

not applicable

9.10.2 XML Schema Element Mapping EReferenceReferenced0001

[To do: todo]

9.10.3 XML Schema Element Mapping EReferenceReferenced0010

[To do: todo]

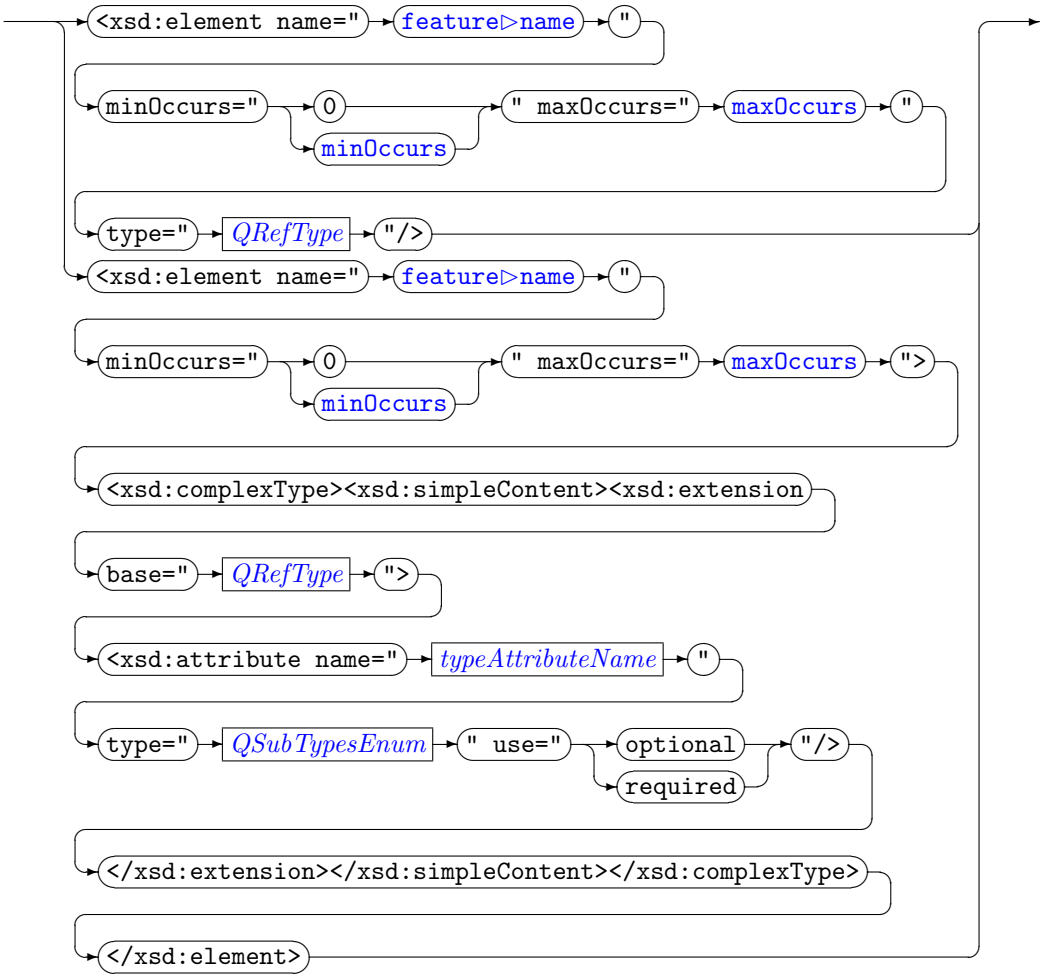
9.10.4 XML Schema Element Mapping EReferenceReferenced0011

[To do: todo]

9.10.5 XML Schema Element Mapping EReferenceReferenced0100

EBNF - Syntax Diagram

SchemaEReferenceReferenced0100



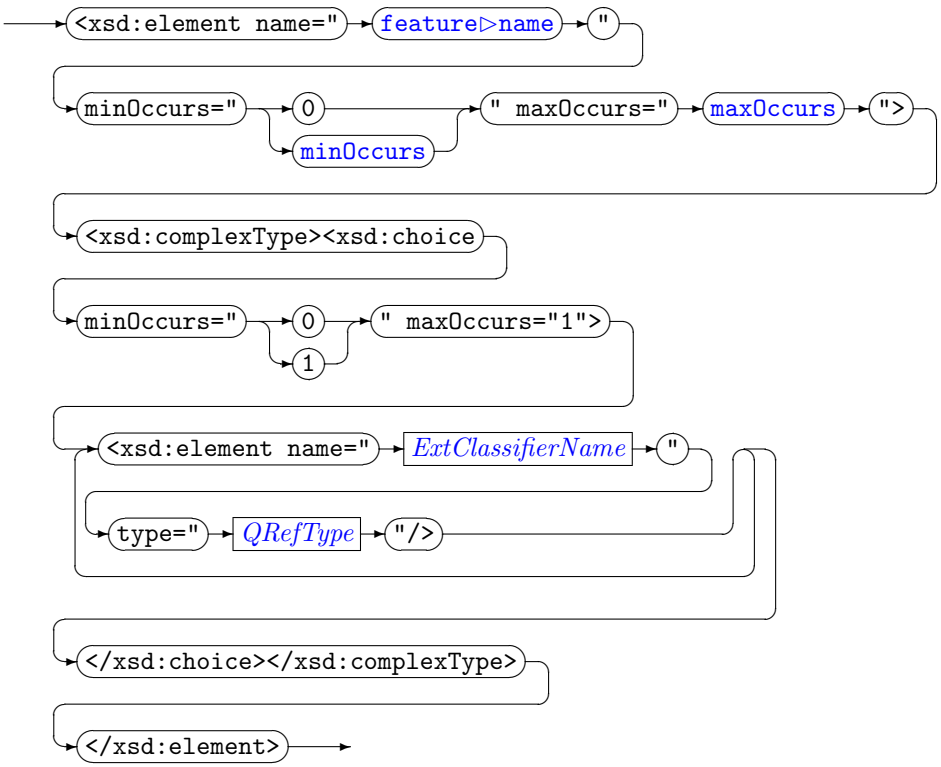
Schema Example

```
<xsd:element name="RELATED-REF"
             minOccurs="0" maxOccurs="unbounded"
             type="node:REF"/>

<xsd:element name="RELATED-REF"
             minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="node:REF">
        <xsd:attribute name="TYPE" type="node:NODE--SUBTYPES-ENUM" use="optional"/>
      </xsd:extension>
   </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

9.10.6 XML Schema Element Mapping EReferenceReferenced0101

EBNF - Syntax Diagram

SchemaEReferenceReferenced0101



Schema Example

```
<xsd:element name="RELATED-REF" minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="1" >
      <xsd:element name="NODE-REFTYPE" type="node:REF"/>
      <xsd:element name="SUB-NODE-REFTYPE" type="node:REF"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.10.7 XML Schema Element Mapping EReferenceReferenced0110

[To do: todo]

9.10.8 XML Schema Element Mapping EReferenceReferenced0111

[To do: todo]

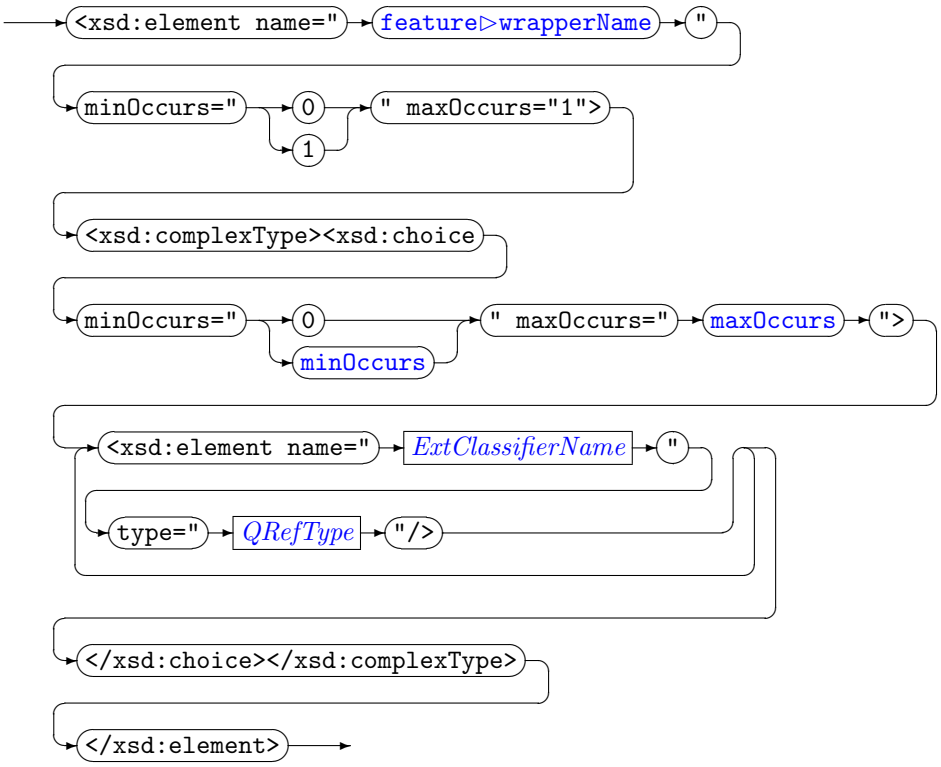
9.10.9 XML Schema Element Mapping EReferenceReferenced1000

[To do: todo]

9.10.10 XML Schema Element Mapping EReferenceReferenced1001

EBNF - Syntax Diagram

SchemaEReferenceReferenced1001



Schema Example

```
<xsd:element name="RELATED-REF" minOccurs="0" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="1" >
      <xsd:element name="NODE-REFTYPE" type="node:REF"/>
      <xsd:element name="SUB-NODE-REFTYPE" type="node:REF"/>
    </xsd:choice>
```

```
  </xsd:complexType>
</xsd:element>
```

9.10.11 XML Schema Element Mapping EReferenceReferenced1010

[To do: todo]

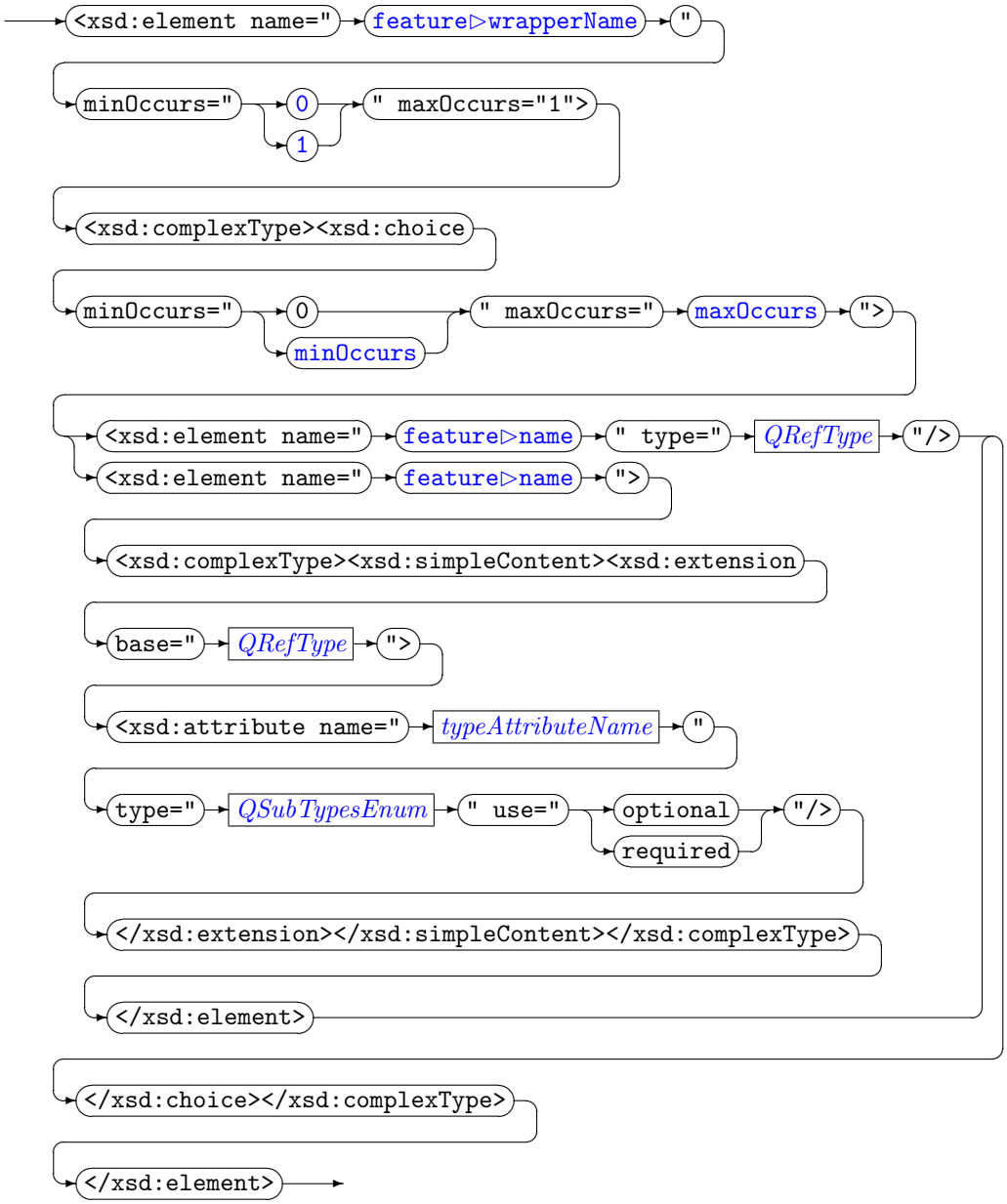
9.10.12 XML Schema Element Mapping EReferenceReferenced1011

[To do: todo]

9.10.13 XML Schema Element Mapping EReferenceReferenced1100

EBNF - Syntax Diagram

SchemaEReferenceReferenced1100



Schema Example

```
<xsd:element name="RELATED-REFS" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="RELATED-REF" type="node:REF"/>
    </xsd:choice>
  </xsd:complexType>
```

```
</xsd:element>

<xsd:element name="RELATED-REFS" minOccurs="0" maxOccurs="1" >
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="RELATED-REF">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="node:REF">
              <xsd:attribute name="TYPE" type="node:NODE--SUBTYPES-ENUM" use="optional"/>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

9.10.14 XML Schema Element Mapping EReferenceReferenced1101

[To do: todo]

9.10.15 XML Schema Element Mapping EReferenceReferenced1110

[To do: todo]

9.10.16 XML Schema Element Mapping EReferenceReferenced1111

[To do: todo]

10 Annex

10.1 XML Persistence Annotations

10.1.1 EPackage

The following annotations are defiined for EPackages:

Table 1: Annotations for EPackage

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		

Table 1: Annotations for EPackage

Key	Valid Values	Description
schemaLocation	<String>	URI that indicates the location of the schema. In most cases this is a publically available internet address. XML processors need to implement a custom resolver in order to load the the schema from the local file system.
externalSchemaLocation	<String>	External name space to schema mapping. The content of this map results in import statements in the XML Schema. The string consists of alternating nsURI and schema locations. E.g. " http://mylang dummy.xsd http://www.w3.org/XML/1998/namespace http://www.w3.org/2001/xml.xsd".
useAttributeGroups	true, false	If true, then reusable attribute groups are produced, which might reduce the size of the XML schema file.
useElementGroups	true, false	If true, then reusable element groups are produced, which might reduce the size of the XML schema file.

In order to reduce the need for adding annotations to many parts of the model, the default values are configurable on the level of the containing EPackage.

Table 2: Annotations for Default Values of EClass

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
default EClass Name	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML names.
default EClass WrapperName	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
default EClass Namespace	#targetNamespace <String>	XML namespace

Table 3: Annotations for Default Values of EDataType

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
default EDataType Name	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML names.
default EDataType WrapperName	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
default EDataType Namespace	#targetNamespace <String>	XML namespace

Table 4: Annotations for Default Values of EAttribute

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
default EAttribute Name	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML names.
default EAttribute WrapperName	##UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
default EAttribute Namespace	##targetNamespace <String>	XML namespace
default EAttribute Kind	unspecified, simple, attribute, attributeWildcard, element, elementWildcard, group	
default EAttribute Many FeatureWrapperElement	true, false	
default EAttribute Many FeatureElement	true, false	
default EAttribute Many ClassifierWrapperElement	true, false	
default EAttribute Many ClassifierElement	true, false	
default EAttribute Single FeatureWrapperElement	true, false	
default EAttribute Single FeatureElement	true, false	
default EAttribute Single ClassifierWrapperElement	true, false	
default EAttribute Single ClassifierElement	true, false	

Table 5: Annotations for Default Values of EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		

Table 5: Annotations for Default Values of EReference

Key	Valid Values	Description
default EReference Namespace	##targetNamespace <String>	XML namespace

Table 6: Annotations for Default Values of containment EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
default EReferenceContained Name	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML names.
default EReferenceContained WrapperName	##UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
default EReferenceContained Many FeatureWrapperElement	true, false	
default EReferenceContained Many FeatureElement	true, false	
default EReferenceContained Many ClassifierWrapperElement	true, false	
default EReferenceContained Many ClassifierElement	true, false	
default EReferenceContained Single FeatureWrapperElement	true, false	
default EReferenceContained Single FeatureElement	true, false	
default EReferenceContained Single ClassifierWrapperElement	true, false	
default EReferenceContained Single ClassifierElement	true, false	
default EReferenceContained TypeAttributeName	<String> xsi:type	

Table 6: Annotations for Default Values of containment EReference

Key	Valid Values	Description
default EReferenceContained TypeDeclaration Strategy	always, ifNeeded	
default EReferenceContained TypeIdentification Strategy	attributeOnly, uriOnly, uriOverwritesAttribute, attributeOverwitesUri	Strategy on where to find information on the contained type.

Table 7: Default Values of non-containment EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
default EReferenceReferenced Name	#UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML names.
default EReferenceReferenced WrapperName	##UpperCaseMapper #IdentityMapper #<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
default EReferenceReferenced Namespace	##targetNamespace <String>	XML namespace
default EReferenceReferenced Kind	attribute, element	
default EReferenceReferenced Many FeatureWrapperElement	true, false	
default EReferenceReferenced Many FeatureElement	true, false	
default EReferenceReferenced Many ClassifierWrapperElement	true, false	
default EReferenceReferenced Many ClassifierElement	true, false	
default EReferenceReferenced Single FeatureWrapperElement	true, false	
default EReferenceReferenced Single FeatureElement	true, false	

Table 7: Default Values of non-containment EReference

Key	Valid Values	Description
default EReferenceReferenced Single ClassifierWrapperElement	true, false	
default EReferenceReferenced Single ClassifierElement	true, false	
default EReferenceReferenced TypeAttributeName	<String> xsi:type	
default EReferenceReferenced TypeDeclaration Strategy	always, ifNeeded	
default EReferenceReferenced TypeIdentification Strategy	attributeOnly, uriOnly, uriOverwritesAttribute, attributeOverwitesUri	Strategy on where to find information on the referenced type.

10.1.2 EClass

Table 8: Annotations of EClass

Key	Valid Values	Description
http:///org/eclipse/emf/ecore/util/ExtendedMetaData		
name	<String>, #<NameMapper>	Name of the EClass in XML. If the value starts with #, then a registered name mapping implementation is selected. If the value is not explicitly given here, the name is calculated by the strategy that is defined by default.EClass.name.
kind		
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
wrapperName	<String># UpperCaseMapper #IdentityMapper #<MapperName>	
xmlGlobalElement	true, false	

10.1.3 EDataType

Table 9: Annotations of EDataType

Key	Valid Values	Description
http:///org/eclipse/emf/ecore/util/ExtendedMetaData		

Table 9: Annotations of EDataType

Key	Valid Values	Description
name	<String>, #<NameMapper>	Name of the EDataType in XML. If the value starts with #, then a registered name mapping implementation is selected. If the value is not explicitly given here, the name is calculated by the strategy that is defined by default.EDataType.name.
baseType		a simple type that is used as base type. See org.eclipse.emf.ecore.xml.type.XMLTypePackage for a list of available predefined base types
fractionDigits		
length		
maxExclusive		
maxInclusive		
maxLength		
minExclusive		
minInclusive		
minLength		
pattern		
totalDigits		
whiteSpace		
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
wrapperName	<String># UpperCaseMapper #IdentityMapper #<MapperName>	
xmlGlobalElement	true, false	

10.1.4 EAttribute

Table 10: Annotations of EAttribute

Key	Valid Values	Description
http:///org/eclipse/emf/ecore/util/ExtendedMetaData		
name	<String>, ##<NameMapper>	Name of the EAttribute in XML. If the value starts with ##, then a registered name mapping implementation is selected. If the value is not explicitly given here, the name is calculated by the strategy that is defined by default.EReference.name.
namespace	<String>, ##targetNamespace	Namespace that is used for representing this EAttribute in XML. ##targetNamespace can be used to refer to the nsURI of the containing package. If the value is not explicitly given here, the value is set to default.EReference.namespace.
kind	unspecified, simple, attribute, attributeWildcard, element, elementWildcard, group	

Table 10: Annotations of EAttribute

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
wrapperName	##UpperCaseMapper ##IdentityMapper ##<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.

Table 11: Additonal Annotations of Single EAttribute

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EAttribute.single .featureWrapperName
featureElement	true, false	for specification of default values please see default.EAttribute.single .featureName
classifierWrapperElement	true, false	for specification of default values please see default.EAttribute.single .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EAttribute.single .classifierName

Table 12: Additional Annotations of Many EAttribute

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EAttribute.many .featureWrapperName
featureElement	true, false	for specification of default values please see default.EAttribute.many .featureName
classifierWrapperElement	true, false	for specification of default values please see default.EAttribute.many .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EAttribute.many .classifierName

10.1.5 Containment EReference

Table 13: Annotations of Containment EReference

Key	Valid Values	Description
http:///org/eclipse/emf/ecore/util/ExtendedMetaData		

Table 13: Annotations of Containment EReference

Key	Valid Values	Description
name	<String>, ##<NameMapper>	Name of the EReference in XML. If the value starts with ##, then a registered name mapping implementation is selected. If the value is not explicitly given here, the name is calculated by the strategy that is derived from default.EReference.name.
namespace	<String>, ##targetNamespace	Namespace that is used for representing this EReference in XML. ##targetNamespace can be used to refer to the nsURI of the containing package. If the value is not explicitly given here, the value is set to default.EReference.namespace.
kind	unspecified, simple, attribute, attributeWildcard, element, elementWildcard, group	
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
wrapperName	##UpperCaseMapper ##IdentityMapper ##<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
classifierNameSuffix	<String>	A suffix that is added to the classifier name if the classifier is used in context of this feature. default: empty
classifierWrapperNameSuffix	<String>	A suffix that is added to the classifier wrapper name if the classifier is used in context of this feature. default: empty
typeAttributeName	xsi:type, <String>	The name of the attribute that is used to identify the type. xsi:type and xmi:type will show up in the XML document only. In case of xsi:type, the XML schema needs implement the generalization dependencies via restriction or extension. default: defaultEReferenceContainedTypeAttributeName
typeIdentificationStrategy	attributeOnly, uriOnly, uriOverwritesAttribute, attributeOverwitesUri	Strategy on where to find information on the referenced type. default: defaultEReferenceContainedTypeIdentificationStrategy
typeDeclarationStrategy	ifNeeded, always	Strategy on when to declare the type. default: defaultEReferenceContainedTypeDeclarationStrategy

Table 14: Additional Annotations of Containment Single EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EReferenceContained.single .featureWrapperName
featureElement	true, false	for specification of default values please see default.EReferenceContained.single .featureName

Table 14: Additional Annotations of Containment Single EReference

Key	Valid Values	Description
classifierWrapperElement	true, false	for specification of default values please see default.EReferenceContained.single .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EReferenceContained.single .classifierName

Table 15: Additional Annotations of Containment Many EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EReferenceContained.many .featureWrapperName
featureElement	true, false	for specification of default values please see default.EReferenceContained.many .featureName
classifierWrapperElement	true, false	for specification of default values please see default.EReferenceContained.many .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EReferenceContained.many .classifierName

10.1.6 Non-Containment EReference

Table 16: Annotations of Non-Containment EReference

Key	Valid Values	Description
http:///org/eclipse/emf/ecore/util/ExtendedMetaData		
name	<String>, ##<NameMapper>	Name of the EReference in XML. If the value starts with ##, then a registered name mapping implementation is selected. If the value is not explicitly given here, the name is calculated by the strategy that is derived from default.EReference.name.
namespace	<String>, ##targetNamespace	Namespace that is used for representing this EReference in XML. ##targetNamespace can be used to refer to the nsURI of the containing package. If the value is not explicitly given here, the value is derived from default.EReference.namespace.
kind	unspecified, simple, attribute, attributeWildcard, element, elementWildcard, group	
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		

Table 16: Annotations of Non-Containment EReference

Key	Valid Values	Description
wrapperName	##UpperCaseMapper ##IdentityMapper ##<MapperName>	Defines the default implementation for mapping model names to XML wrapper names.
classifierNameSuffix	<String>	A suffix that is added to the classifier name if the classifier is used in context of this feature. default: empty
classifierWrapperNameSuffix	<String>	A suffix that is added to the classifier wrapper name if the classifier is used in context of this feature. default: empty
typeAttributeName	xsi:type, <String>	The name of the attribute that is used to identify the type. xsi:type and xmi:type will show up in the XML document only. In case of xsi:type, the XML schema needs implement the generalization dependencies via restriction or extension. default: defaultEReferenceReferencedTypeAttributeName
typeIdentificationStrategy	attributeOnly, uriOnly, uriOverwritesAttribute, attributeOverwitesUri	Strategy on where to find information on the referenced type. default: defaultEReferenceReferencedTypeIdentificationStrategy
typeDeclarationStrategy	ifNeeded, always	Strategy on when to declare the type. default: defaultEReferenceReferencedTypeDeclarationStrategy

Table 17: Additional Annotations of Non-Containment Single EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EReferenceReferenced.single .featureWrapperName
featureElement	true, false	for specification of default values please see default.EReferenceReferenced.single .featureName
classifierWrapperElement	true, false	for specification of default values please see default.EReferenceReferenced.single .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EReferenceReferenced.single .classifierName

Table 18: Additional Annotations of Non-Containment Many EReference

Key	Valid Values	Description
http:///org/eclipse/sphinx/emf/serialization/XMLPersistenceMappingExtendedMetaData		
featureWrapperElement	true, false	for specification of default values please see default.EReferenceReferenced.many .featureWrapperName

Table 18: Additional Annotations of Non-Containment Many EReference

Key	Valid Values	Description
featureElement	true, false	for specification of default values please see default.EReferenceReferenced.many .featureName
classifierWrapperElement	true, false	for specification of default values please see default.EReferenceReferenced.many .classifierWrapperName
classifierElement	true, false	for specification of default values please see default.EReferenceReferenced.many .classifierName

10.1.7 Overview XML Persistence Mapping

References

[1] Frank Budinsky. *XML Schema to Ecore Mapping*. 2004. URL: http://www.eclipse.org/modeling/emf/docs/overviews/XMLSchemaToEcoreMapping.pdf.

[2] Ed Merks. *The Art of Java Performance Tuning*. EclipseCon Europe. 2012. URL: http://wiki.eclipse.org/images/c/ca/JavaPerformanceTuning.pdf.

[3] Object Management Group (OMG). *OMG MOF 2 XMI Mapping Specification, version 2.4.1*. 2013.

[4] Object Management Group (OMG). *OMG XML Metadata Interchange (XMI) Specification, version 1.2*. 2002.

[5] David Steinberg, Marcelo Paternostro, and Kenn Hussey. *Getting the Most out of Your Models, Performance and Extensibility with EMF*. 2010. URL: http://de.slideshare.net/dmsteinberg/eclipsecon-2010-getting-the-most-out-of-your-models-performance-and-extensibility-with-emf.

[6] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. URL: http://www.w3.org/TR/xml/.

[7] W3C. *Extensible Markup Language (XML) 1.1 (Second Edition)*. 2006. URL: http://www.w3.org/TR/xml11/.

[8] W3C. *XML Schema Part 1: Structures Second Edition*. 2004. URL: http://www.w3.org/TR/xmlschema-1/.

[9] W3C. *XML Schema Part 2: Datatypes*. 2001. URL: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

Index