OWL Web Ontology Language XML Presentation Syntax

Masahiro Hori, Jérôme Euzenat, Peter Patel-Schneider

W3C Note

OWL Web Ontology Language
XML Presentation Syntax

W3C Note 11 June 2003

New Version Available: OWL 2 (Document Status Update, 12 November 2009)

The OWL Working Group has produced a W3C Recommendation for a new version of OWL which adds features to this 2004 version, while remaining compatible. Please see OWL 2 Document Overview for an introduction to OWL 2 and a guide to the OWL 2 document set.

This version:
http://www.w3.org/TR/2003/NOTE-owl-xmlsyntax-20030611/
Latest version:
http://www.w3.org/TR/owl-xmlsyntax/
Authors:
Masahiro Hori, Kansai University (formerly IBM Tokyo Research)
Jérôme Euzenat, INRIA Rhône-Alpes
Peter F. Patel-Schneider, Bell Labs Research, Lucent Technologies

Abstract

This document specifies XML presentation syntax for OWL, which is defined as a dialect similar to OWL Abstract Syntax [OWL Semantics]. It is not intended to be a normative specification. Instead, it represents a suggestion of one possible XML presentation syntax for OWL.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This document has been produced as part of the W3C Semantic Web Activity (Activity Statement) following the procedures set out for the W3C Process. The goals of the Web Ontology working group are discussed in the Web Ontology Working Group charter.

This version is based on the Working Drafts dated on 31 March: OWL Web Ontology Language Semantics and Abstract Syntax, OWL Web Ontology Language Reference, and OWL Web Ontology Language Guide.

There are no patent disclosures related to this work at the time of this writing.

The authors welcome comments on this document, but does not guarantee a reply or any further action. Please send comments on this draft to public-webont-comments@w3.org; public archives are available. This document may be updated or added to based on implementation experience, but no commitment is made by the W3C, or any of its members, regarding future updates.

This document is a NOTE made available by the W3C for discussion only. Publication of this Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members. A list of current W3C technical reports and publications, including Working Drafts and Notes, can be found at http://www.w3.org/TR/.

Table of Contents

---

1 Introduction

This document specifies XML presentation syntax for OWL, which is defined as a dialect similar to OWL Abstract Syntax [OWL Semantics]. The OWL language provides three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full. This document provides XML Schemas for XML presentation syntax corresponding to the three sublanguages. Each of the sublanguages is an extension of its simpler predecessor, and the following relations hold but the inverses do not.

- Every document valid against the XML Schema of OWL Lite is valid against the XML Schema of OWL DL.
- Every document valid against the XML Schema of OWL DL is valid against the XML Schema of OWL Full.

Three Schemas are defined for the sublanguages: OWL Lite (owl1-lite.xsd), OWL DL (owl1-dl.xsd), and OWL Full (owl1-full.xsd). The definitions of these Schemas are given in Appendix A.

1.1 OWL Document Structure

The root element of OWL documents for the XML presentation syntax must be **`Ontology`** element (2.1), and an OWL document further consists of optional header elements (2.2) as well as any number of classes (2.3), properties (2.4), and individuals (2.5).

Example 1_1-1 (L,D,F)

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>
<owlx:Ontology owlx:name="http://www.example.org/wine"
  xmlns:owlx="http://www.w3.org/2003/05/owl-xml">

  <!--
    Contains optional header elements as well as
    any number of classes, properties, and individuals.
  -->

</owlx:Ontology>
```

Attribute values are frequently used for referring to identifiers in OWL. Since attribute values are not namespace sensitive, they may be written down in their fully expanded form (e.g., `http://www.example.org/owl/wine#merlot`). Alternatively, XML entities can be used as abbreviations. In the XML presentation syntax, XML Schema datatypes (2.6) are referred to by `owlx:datatype` attributes, and an XML entity `&xsd;` can be used for readability.

1.2 Notational Conventions

A number of namespace prefixes are used in this document as listed below. Note that the choice of any namespace prefix is arbitrary, and not semantically significant.

Prefix	Namespace	Notes
owlx	"http://www.w3.org/2003/05/owl-xml"	The namespace of the XML presentation syntax of OWL defined by this document
owl	"http://www.w3.org/2002/07/owl"	The namespace of OWL in RDF/XML syntax

xsd	"http://www.w3.org/2001/XMLSchema"	The namespace of the XML Schema [XMLSchema-1]

2. Element Reference

All the elements and their attributes and element contents in the XML syntax are listed in this section. In order to indicate partial availability for OWL sublanguages, subscript enclosed within parentheses [e.g., `(L)` and `(D,F)`] may be appended to either element names, attribute names, or labels for element contents (i.e., *`Content:`*). For example, `EnumeratedClass`$_{(D,F)}$ means that `EnumeratedClass` element is only available for OWL DL and OWL Full. Note that all the elements, attributes, and element contents without subscript in parentheses means that those constructs are available for all the three OWL sublanguages.

Examples and their brief explanations of OWL statements in this section are derived from the Reference document [OWL Reference], and more examples in Appendix B are created on the basis of the Guide document [OWL Guide]. In addition, element index (3.1) gives an alphabetical list of elements in the XML presentation syntax, and cross reference (3.2) shows the correspondence between constructs in RDF/XML syntax and elements of XML presentation syntax. In Appendix C, an XSLT stylesheet is given for the transformation from the XML presentation syntax toward RDF/XML syntax of OWL.

2.1 The Root Element

element **Ontology**

```
<Ontology
  name = xsd:anyURI
>
  Content: (VersionInfo | PriorVersion | BackwardCompatibleWith |
            IncompatibleWith | Imports | Annotation |
            Class[axiom] | EnumeratedClass(D,F) |
            SubClassOf(D,F) | EquivalentClasses | DisjointClasses(D,F) |
            DatatypeProperty | ObjectProperty |
            SubPropertyOf | EquivalentProperties |
            Individual[axiom] | SameIndividual | DifferentIndividuals)*
</Ontology>
```

Attribute:	**name** - refers to a name of this ontology, which is the base URI of this element.
Note:	This is the root element of OWL documents in the XML presentation syntax, while `rdf:RDF` is used as the document root for OWL in RDF/XML.
See also:	owl:Ontology used as a construct for the header component in [OWL Reference]

2.2 Header Elements

The root element `Ontology` may optionally contain, in any order, any number of header elements for versioning import and/or annotation statements. In the presentation syntax, there is no header element in contrast to the RDF/XML syntax in which `owl:Ontology` is provided as a construct for the header component.

A sample ontology header could look like this (see also 3 in [OWL Reference]):

Example 2_2-1 (L,D,F)

```
<owlx:VersionInfo>2003/02/24 22:37:39</owlx:VersionInfo>
<owlx:Annotation>
  <owlx:Documentation>An example OWL ontology</owlx:Documentation>
</owlx:Annotation>
<owlx:PriorVersion owlx:ontology="http://www.example.org/wine-112102.owl" />
<owlx:Imports owlx:ontology="http://www.example.org/food.owl"/>
<owlx:Annotation>
  <owlx:Label>Wine Ontology</owlx:Label>
</owlx:Annotation>
```

element **VersionInfo**

```
<VersionInfo>
  Content: (#CDATA)
</VersionInfo>
```

Parents:	Ontology
Note:	This element contains a string giving information about this version such as RCS/CVS keywords.
See also:	owl:versionInfo in [OWL Reference]

element **PriorVersion**

```
<PriorVersion
  ontology = xsd:anyURI {required}
>
  Content: (##empty)
</PriorVersion>
```

Attribute:	**ontology** - a reference to a prior ontology
Parents:	Ontology
Note:	This element contains a reference to another ontology that identifies the specified ontology as a prior version of the containing ontology.
See also:	owl:priorVersion in [OWL Reference]

element **BackwardCompatibleWith**

```
<BackwardCompatibleWith
  ontology = xsd:anyURI {required}
>
  Content: (##empty)
</BackwardCompatibleWith>
```

Attribute:	**ontology** - a reference to a backward compatible ontology
Parents:	Ontology
Note:	This element contains a reference to another ontology that identifies the specified ontology as a prior version of the containing ontology, and further indicates that it is backward compatible with it.
See also:	owl:backwardCompatibleWith in [OWL Reference]

element **IncompatibleWith**

```
<IncompatibleWith
  ontology = xsd:anyURI {required}
>
  Content: (##empty)
</IncompatibleWith>
```

Attribute: **ontology** - a reference to an incompatible ontology

Parents: Ontology

Note: This element contains a reference to another ontology that indicates that the containing ontology is a later version of the referenced ontology, but is not backward compatible with it.

See also: owl:incompatibleWith in [OWL Reference]

element **Imports**

```
<Imports
  ontology = xsd:anyURI {required}
>
  Content: (##empty)
</Imports>
```

Attribute: **ontology** - a reference to another ontology

Parents: Ontology

Note: This element refers to another OWL ontology containing definitions, whose meaning is considered to be part of the meaning of the importing ontology.

See also: owl:imports in [OWL Reference]

element **Annotation**

```
<Annotation>
  Content: (Label | Documentation | xsd:any )*
</Annotation>
```

Parents: Ontology, Class[axiom], EnumeratedClass(D,F), DatatypeProperty, ObjectProperty, Individual[axiom]

Note: This element is provided as a place holder for various ways of annotations such as `Label` and `Documentation`. In particular, this element can take a wildcard element (`xsd:any`), which can be in any namespace other than the target namespace 'owlx' for the XML presentation syntax Schema (http://www.w3.org/2003/05/owl-xml), since `namespace` attribute value of the `xsd:any` element is specified as `##other` in the Schema declaration. In addition, according to the declaration `processContents="lax"` in the `xsd:any`, the wildcard element allows an XML Schema processor to validate elements for which it can find namespace declarations and raise errors if they are invalid. However, the processor will not report errors on the elements for which it does not find declarations.

element **Label**

```
<Label
  xml:lang = xsd:language
>
  Content: (#CDATA)
</Label>
```

Attribute: **xml:lang** - specifies the language used in the content

Parents: Annotation

Note: This element provides a human-readable version name of an annotated element. In RDF/XML syntax, this element is mapped to `rdfs:label` in [RDF Schema].

element **Documentation**

```
<Documentation
  xml:lang = xsd:language
```

```
  >
    Content: (#CDATA)
  </Documentation>
```

Attribute: **xml:lang** - specifies the language used in the content

Parents: Annotation

Note: This element provides a human-readable description of an annotated element. In RDF/XML syntax, this element is mapped to `rdfs:comment` in [RDF Schema].

2.3 Classes

Classes provide an abstraction mechanism for grouping resources with similar characteristics. OWL classes are described through "class descriptions", which can be combined into "class axioms". "Axiom" is a formal term, and may be called "definition" informally.

element **Class**[axiom]

```
  <Class
    name = xsd:anyURI {required}
    complete = xsd:boolean {required}
    deprecated = xsd:boolean
  >
    Content: (Annotation*, description*)
  </Class>
```

Attribute: **name** - a reference to a name of this Class
complete - the modality is complete if true, otherwise partial. The default value is false.
deprecated - this class is deprecated if true

Parents: Ontology

Note: This element contains non-empty sequence of descriptions, which are the basic building blocks of class axioms.

See also: owl:Class in [OWL Reference]

2.3.1 Class descriptions

A class description is the term used in [OWL Reference] [OWL Semantics], and used as a basic building block of class axioms (or called class definitions informally).

Model group ***description***

```
  Content: (Class[ID] | DataRestriction | ObjectRestriction | UnionOf(D,F) |
            IntersectionOf(D,F) | ComplementOf(D,F) | OneOf[object](D,F))
```

Parents: Class[axiom], EquivalentClasses, DisjointClasses(D,F), sub(D,F), super(D,F), UnionOf(D,F), IntersectionOf(D,F), ComplementOf(D,F), domain(D,F), ObjectProperty/range(D,F), ObjectRestriction/allValuesFrom(D,F), ObjectRestriction/someValuesFrom(D,F), type

Note: In the XML presentation syntax, a class description is represented as a model group definition of XML Schema [XMLSchema-1], which can be used as a reusable fragment of a content model. This model group allows to not only describe a class through a name (i.e., a class identifier), but also define an

anonymous class by means of property restriction, union, intersection, complement, and enumeration.

See also: 4.1 in [OWL Reference]

element **Class**[ID]

```
<Class
  name = xsd:anyURI {required}
>
  Content: (##empty)
</Class>
```

Attribute: **name** - a reference to a class name

Parents: *description*, EquivalentClasses(L), domain(L), ObjectProperty/range(L)

Note: This element describes a class through a name.

Example 2_3-1 (L,D,F)

```
<owlx:Class owlx:name="Student" owlx:complete="false">
  <owlx:Annotation>
    <owlx:Label xml:lang="nl">student</owlx:Label>
    <owlx:Label xml:lang="it">studente</owlx:Label>
    <owlx:Label xml:lang="es">estudiante</owlx:Label>
    <owlx:Documentation xml:lang="en">Undergraduates students</owlx:Documentation>
  </owlx:Annotation>
  <owlx:Class owlx:name="Person" />
</owlx:Class>
```

2.3.2 Property restrictions

A property restriction defines an anonymous class, namely a class of all individuals that satisfy the restriction. The XML syntax of OWL distinguishes two types of property restrictions: DataRestriction (restriction on properties for which the range value is a data literal) and ObjectRestriction (restriction on properties for which the range value is an individual).

Example 2_3-2 (D,F)

```
<owlx:Class owlx:name="Student" owlx:complete="false">
  <owlx:Class owlx:name="Person" />

  <owlx:DataRestriction owlx:property="studentID">
    <owlx:cardinality owlx:value="1" />
    <owlx:allValuesFrom owlx:datatype="&xsd;integer" />
  </owlx:DataRestriction>

  <owlx:DataRestriction owlx:property="grade">
    <owlx:cardinality owlx:value="1" />
    <owlx:allValuesFrom>
      <owlx:OneOf>
        <owlx:DataValue owlx:datatype="&xsd;integer">1</owlx:DataValue>
        <owlx:DataValue owlx:datatype="&xsd;integer">2</owlx:DataValue>
        <owlx:DataValue owlx:datatype="&xsd;integer">3</owlx:DataValue>
        <owlx:DataValue owlx:datatype="&xsd;integer">4</owlx:DataValue>
      </owlx:OneOf>
    </owlx:allValuesFrom>
  </owlx:DataRestriction>

  <owlx:ObjectRestriction owlx:property="schoolYear">
    <owlx:cardinality owlx:value="1" />
    <owlx:allValuesFrom>
```

```
      <owlx:OneOf>
        <owlx:Individual owlx:name="Freshman" />
        <owlx:Individual owlx:name="Sophomore" />
        <owlx:Individual owlx:name="Junior" />
        <owlx:Individual owlx:name="Senior" />
      </owlx:OneOf>
    </owlx:allValuesFrom>
  </owlx:ObjectRestriction>

  <owlx:ObjectRestriction owlx:property="advisor">
    <owlx:minCardinality owlx:value="1" />
    <owlx:allValuesFrom owlx:class="Person" />
    <owlx:someValuesFrom>
      <owlx:Class owlx:name="Professor" />
      <owlx:Class owlx:name="AssociateProfessor" />
    </owlx:someValuesFrom>
  </owlx:ObjectRestriction>

</owlx:Class>
```

element **DataRestriction**

```
<DataRestriction
  property = xsd:anyURI {required}
>
  Content(L): ( (minCardinality | maxCardinality | cardinality |
                 allValuesFrom | someValuesFrom) )
  Content(D,F): ( (minCardinality | maxCardinality | cardinality |
                   allValuesFrom | someValuesFrom | hasValue)+ )
</DataRestriction>
```

Attribute:	**property** - a reference to a property name
Parents:	*description*
Note:	This element declares restrictions on DatatypeProperty (properties for which the range value is a data literal). The restrictions must have exactly one content element for OWL Lite, while they must have at least one content element for OWL DL and Full.
See also:	owl:Restriction, owl:minCardinality, owl:maxCardinality, owl:cardinality, owl:allValuesFrom, owl:someValuesFrom, and owl:hasValue in [OWL Reference]

element **minCardinality**

```
<minCardinality
  value(L)   = either '0' or '1' {required}
  value(D,F) = xsd:nonNegativeInteger {required}
>
  Content: ( ##empty )
</minCardinality>
```

Attribute:	**value** - a minimum cardinality value
Parents:	DataRestriction, ObjectRestriction
Note:	This element declares restriction on a minimum cardinality value.
See also:	owl:minCardinality in [OWL Reference]

element **maxCardinality**

```
<maxCardinality
  value(L)   = either '0' or '1' {required}
  value(D,F) = xsd:nonNegativeInteger {required}
>
  Content: ( ##empty )
</maxCardinality>
```

Attribute: **value** - a maximum cardinality value

Parents: DataRestriction, ObjectRestriction

Note: This element declares restriction on a maximum cardinality value.

See also: owl:maxCardinality in [OWL Reference]

element **cardinality**

```
<cardinality
  value(L)   = either '0' or '1' {required}
  value(D,F) = xsd:nonNegativeInteger {required}
>
  Content: ( ##empty )
</cardinality>
```

Attribute: **value** - a cardinality value

Parents: DataRestriction, ObjectRestriction

Note: This element declares restriction on a cardinality value.

See also: owl:cardinality in [OWL Reference]

element DataRestriction/**allValuesFrom**

```
<allValuesFrom
  datatype = xsd:anyURI
>
  Content(D,F): ( OneOf[data]? )
</allValuesFrom>
```

Attribute: **datatype** - a reference to a datatype

Parents: DataRestriction

Note: This element specifies a class of all individuals for which all range values of the property under consideration are data values within the specified data range.

See also: owl:allValuesFrom in [OWL Reference]

element DataRestriction/**someValuesFrom**

```
<someValuesFrom
  datatype = xsd:anyURI
>
  Content(D,F): ( OneOf[data]? )
</someValuesFrom>
```

Attribute: **datatype** - a reference to a datatype

Parents: DataRestriction

Note: This element specifies a class of all individuals for which at least one value of the property concerned is a data value in the data range.

See also: owl:someValuesFrom in [OWL Reference]

element DataRestriction/**hasValue**$_{(D,F)}$

```
<hasValue>
  Content: ( xsd:anySimpleType )
</hasValue>
```

Parents: DataRestriction

Note: This element specifies a type of data value for which the property concerned has at least one value.

See also: owl:hasValue in [OWL Reference]

element **ObjectRestriction**

```
<ObjectRestriction
  property = xsd:anyURI {required}
>
  Content(L): ( (minCardinality | maxCardinality | cardinality |
                 allValuesFrom | someValuesFrom) )
  Content(D,F): ( (minCardinality | maxCardinality | cardinality |
                   allValuesFrom | someValuesFrom | hasValue)+ )
</ObjectRestriction>
```

Attribute: **property** - a reference to a property name

Parents: *description*

Note: This element declares restrictions on ObjectProperty (properties for which the range value is an individual). The restrictions must have exactly one content element for OWL Lite, while they must have at least one content element for OWL DL and Full.

See also: owl:Restriction, owl:minCardinality, owl:maxCardinality, owl:cardinality, owl:allValuesFrom, owl:someValuesFrom, and owl:hasValue in [OWL Reference]

element ObjectRestriction/**allValuesFrom**

```
<allValuesFrom
  class = xsd:anyURI
>
  Content: ( description(D,F)* )
</allValuesFrom>
```

Attribute: **class** - a reference to a class name

Parents: ObjectRestriction

Note: This element specifies a class of all individuals for which all range values of the property under consideration are members of the class extension of the class description.

See also: owl:allValuesFrom in [OWL Reference]

The example below describes a class of all individuals for which the "hasParent" property only has range values of class "Human" (see also owl:allValuesFrom in [OWL Reference]).

Example 2_3-3 (L,D,F)

```
<owlx:Class owlx:name="Child1" owlx:complete="false">
  <owlx:ObjectRestriction owlx:property="#hasParent">
    <owlx:allValuesFrom owlx:class="#Human" />
  </owlx:ObjectRestriction>
</owlx:Class>
```

element ObjectRestriction/**someValuesFrom**

```
<someValuesFrom
  class = xsd:anyURI
>
  Content: ( description(D,F)* )
</someValuesFrom>
```

Attribute: **class** - a reference to a class name

Parents: ObjectRestriction

Note:	This element specifies a class of all individuals for which at least one value of the property concerned is an instance of the class description.
See also:	owl:someValuesFrom in [OWL Reference]

The following example defines a class of individuals which have at least one parent who is a physician (see also owl:someValuesFrom in [OWL Reference]):

Example 2_3-4 (L,D,F)

```
<owlx:Class owlx:name="Child2" owlx:complete="false">
  <owlx:ObjectRestriction owlx:property="#hasParent">
    <owlx:someValuesFrom owlx:class="#Physician" />
  </owlx:ObjectRestriction>
</owlx:Class>
```

element ObjectRestriction/**hasValue**(D,F)

```
<hasValue
  name = xsd:anyURI {required}
>
  Content: ( ##empty )
</hasValue>
```

Attribute:	**name** - a reference to a class name
Parents:	ObjectRestriction
Note:	This element specifies a class of all individuals for which the property concerned has at least one value.
See also:	owl:hasValue in [OWL Reference]

The following example describes the class of individuals who have the individual referred to as "Clinton" as their parent (see also owl:hasValue in [OWL Reference]):

Example 2_3-5 (L,D,F)

```
<owlx:Class owlx:name="Child3" owlx:complete="false">
  <owlx:ObjectRestriction owlx:property="#hasParent">
    <owlx:hasValue owlx:name="#Clinton" />
  </owlx:ObjectRestriction>
</owlx:Class>
```

2.3.3 Enumeration of individuals

element **OneOf** [object](D,F)

```
<OneOf>
  Content: ( Individual[ID]* )
</OneOf>
```

Parents:	*description*
Note:	This element contains the exactly enumerated individuals. On the other hand, enumeration of data values is done by OneOf[data].
See also:	owl:oneOf in [OWL Reference], and EnumeratedClass in this document

For example, the following `OneOf` statement defines a class of all continents (see also Enumeration in [OWL Reference]):

Example 2_3-6 (D,F)

```
<owlx:Class owlx:name="AllContinents" owlx:complete="true">
  <owlx:OneOf>
    <owlx:Individual owlx:name="#Eurasia" />
    <owlx:Individual owlx:name="#Africa" />
    <owlx:Individual owlx:name="#NorthAmerica" />
    <owlx:Individual owlx:name="#SouthAmerica" />
    <owlx:Individual owlx:name="#Australia" />
    <owlx:Individual owlx:name="#Antarctica" />
  </owlx:OneOf>
</owlx:Class>
```

A statement of the form `<owlx:Individual owlx:name="..."/>` refers to some individual (remember: all individuals are by definition instances of `owlx:Individual`).

In the section on datatypes we will see another use of the `OneOf` element, namely to define an enumeration of data values.

element **EnumeratedClass**(D,F)

```
<EnumeratedClass
  name = xsd:anyURI {required}
  deprecated = xsd:boolean
>
  Content: (Annotation*, Individual[ID]*)
</EnumeratedClass>
```

Attribute:	**name** - a reference to a name of this class **deprecated** - this class is deprecated if true
Parents:	`Ontology`
Note:	This element contains the exactly enumerated individuals.
See also:	owl:oneOf in [OWL Reference], and `OneOf`[object] in this document

`EnumeratedClass` can also be used for the `OneOf` enumeration, and the enumeration of all continents can be described as follows (see also Enumeration in [OWL Reference]):

Example 2_3-7 (D,F)

```
<owlx:EnumeratedClass owlx:name="AllContinents">
  <owlx:Individual owlx:name="#Eurasia" />
  <owlx:Individual owlx:name="#Africa" />
  <owlx:Individual owlx:name="#NorthAmerica" />
  <owlx:Individual owlx:name="#SouthAmerica" />
  <owlx:Individual owlx:name="#Australia" />
  <owlx:Individual owlx:name="#Antarctica" />
</owlx:EnumeratedClass>
```

2.3.4 Boolean combination

element **IntersectionOf**(D,F)

```
<IntersectionOf
  class = xsd:anyURI
>
```

```
  Content: ( description* )
</IntersectionOf>
```

Attribute: **class** - a reference to a class name

Parents: *description*

Note: This element specifies a class for which the class extension contains precisely those individuals that are members of the class extension of all class descriptions in the range list.

See also: owl:intersectionOf in [OWL Reference]

In the example below, the range of the intersection statement is a list of two class descriptions, namely two enumerations, both describing a class with two individuals. The resulting intersection is a class with one individual, namely "Tosca" since this is the only individual that is common to both enumerations (see also owl:intersectionOf in [OWL Reference]).

Example 2_3-8 (D,F)

```
<owlx:Class owlx:name="IntersectionOf-eg" owlx:complete="true">
  <owlx:IntersectionOf>
    <owlx:OneOf>
      <owlx:Individual owlx:name="#Tosca" />
      <owlx:Individual owlx:name="#Salome" />
    </owlx:OneOf>
    <owlx:OneOf>
      <owlx:Individual owlx:name="#Turandot" />
      <owlx:Individual owlx:name="#Tosca" />
    </owlx:OneOf>
  </owlx:IntersectionOf>
</owlx:Class>
```

element **UnionOf**$_{(D,F)}$

```
<UnionOf
  class = xsd:anyURI
>
  Content: ( description* )
</UnionOf>
```

Attribute: **class** - a reference to a class name

Parents: *description*

Note: This element specifies an anonymous class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the range list.

See also: owl:unionOf in [OWL Reference]

The example below describes a class for which the class extension contains three individuals, namely "Tosca", "Salome", and "Turandot" assuming they are all different (see also owl:unionOf in [OWL Reference]).

Example 2_3-9 (D,F)

```
<owlx:Class owlx:name="UnionOf-eg" owlx:complete="true">
  <owlx:UnionOf>
    <owlx:OneOf>
      <owlx:Individual owlx:name="#Tosca" />
      <owlx:Individual owlx:name="#Salome" />
```

```
    </owlx:OneOf>
    <owlx:OneOf>
      <owlx:Individual owlx:name="#Turandot" />
      <owlx:Individual owlx:name="#Tosca" />
    </owlx:OneOf>
  </owlx:UnionOf>
</owlx:Class>
```

element **ComplementOf**(D,F)

```
<ComplementOf>
  Content: ( description )
</ComplementOf>
```

Parents:	*description*
Note:	This element specifies a class for which the class extension contains exactly those individuals that do not belong to the class extension of the range class.
See also:	owl:complementOf in [OWL Reference]

For example, the expression "neither meat nor fish" could be written as follows by using `ComplementOf` element (see also owl:complementOf in [OWL Reference]).

Example 2_3-10 (D,F)

```
<owlx:Class owlx:name="ComplementOf-eg" owlx:complete="true">
  <owlx:ComplementOf>
    <owlx:UnionOf>
      <owlx:Class owlx:name="#Meat" />
      <owlx:Class owlx:name="#Fish" />
    </owlx:UnionOf>
  </owlx:ComplementOf>
</owlx:Class>
```

2.3.5 Class relationships

element **SubClassOf**(D,F)

```
<SubClassOf>
  Content: ( sub, super )
</SubClassOf>
```

Parents:	Ontology
Note:	This element allows one to say that the class extension of a class description is a subset of the class extension of another class description.
See also:	rdfs:subClassOf in [OWL Reference]

element **sub**(D,F)

```
<sub>
  Content: ( description )
</sub>
```

Parents: SubClassOf

element **super**(D,F)

```
<super>
  Content: ( description )
```

```
</super>
```

Parents: SubClassOf

For example, the following class axiom declares a subclass relation between two classes that are described through their names ("Opera" and "MusicalWork"). Subclass relations provide necessary conditions for belonging to a class. In this case, to be an opera the individual also needs to be a musical work (see also rdfs:subClassOf in [OWL Reference]).

Example 2_3-11 (L,D,F)

```
<owlx:Class owlx:name="Opera" owlx:complete="false">
  <owlx:Class owlx:name="#MusicalWork" />
</owlx:Class>
```

For any class there may be any number of subClassOf axioms. For example, it is possible to add the following axiom about the `Opera` class.

Example 2_3-12 (D,F)

```
<owlx:SubClassOf>
  <owlx:sub>
    <owlx:Class owlx:name="Opera" />
  </owlx:sub>
  <owlx:super>
    <owlx:ObjectRestriction owlx:property="#hasLibrettist">
      <owlx:minCardinality owlx:value="1" />
    </owlx:ObjectRestriction>
  </owlx:super>
</owlx:SubClassOf>
```

This class axiom contains a property restriction. The example states that `Opera` is a subclass of an anonymous OWL class that has as its class extension the set of all individuals for which the property `hasLibrettist` has at least one value. Thus, operas should have at least one librettist.

element **EquivalentClasses**

```
<EquivalentClasses>
  Content(L): ( Class[ID], Class[ID]+ )
  Content(D,F): ( description, description+ )
</EquivalentClasses>
```

Parents:	Ontology
Note:	This element asserts that two or more classes have the same class extension (i.e., class extensions contain exactly the same set of individuals).
See also:	owl:equivalentClass in [OWL Reference]

In its simplest form, an `EquivalentClass` axiom states the equivalence (in terms of their class extension) of two named classes. For example:

Example 2_3-13 (L,D,F)

```
<owlx:EquivalentClasses>
  <owlx:Class owlx:name="US_President" />
  <owlx:Class owlx:name="PrincipalResidentOfWhiteHouse" />
</owlx:EquivalentClasses>
```

element **DisjointClasses**$_{(D,F)}$

```
<DisjointClasses>
  Content: ( description, description+ )
</DisjointClasses>
```

Parents:	Ontology
Note:	This element asserts that the class extensions of the two or more class descriptions involved have no individuals in common.
See also:	owl:disjointWith in [OWL Reference]

Declaring two classes to be disjoint is a partial definition: it imposes a necessary but not sufficient condition on the class. A popular example of class disjointness is given below:

Example 2_3-14 (D,F)

```
<owlx:DisjointClasses>
  <owlx:Class owlx:name="#Man" />
  <owlx:Class owlx:name="#Woman" />
</owlx:DisjointClasses>
```

The following example shows a common use of class disjointness in subclass hierarchies. The next example indicates that `MusicDrama` is either an `opera`, an `Operetta`, or a `Musical` (i.e., the subclass partitioning is complete).

Example 2_3-15 (D,F)

```
<owlx:Class owlx:name="MusicDrama" owlx:complete="true">
  <owlx:UnionOf>
    <owlx:Class owlx:name="#Opera" />
    <owlx:Class owlx:name="#Operetta" />
    <owlx:Class owlx:name="#Musical" />
  </owlx:UnionOf>
</owlx:Class>
```

In the examples below, individuals belonging to one subclass such as `Opera` cannot belong to another subclass such as `Musical` (disjoint or non-overlapping subclasses).

Example 2_3-16 (D,F)

```
<owlx:Class owlx:name="#Opera" owlx:complete="false">
  <owlx:Class owlx:name="#MusicDrama" />
</owlx:Class>

<owlx:Class owlx:name="#Operetta" owlx:complete="false">
  <owlx:Class owlx:name="#MusicDrama" />
</owlx:Class>

<owlx:Class owlx:name="#Musical" owlx:complete="false">
  <owlx:Class owlx:name="#MusicDrama" />
</owlx:Class>

<owlx:DisjointClasses>
  <owlx:Class owlx:name="#Opera" />
  <owlx:Class owlx:name="#Operetta" />
  <owlx:Class owlx:name="#Musical" />
</owlx:DisjointClasses>
```

In the above examples, `DisjointWith` elements were used together with `UnionOf`, in order to define a set of mutually disjoint and complete subclasses of a superclass (see also owl:disjointWith in [OWL Reference]).

2.4 Properties

OWL distinguishes between two types of properties: `DatatypeProperty` and `ObjectProperty`.

element **DatatypeProperty**

```
<DatatypeProperty
  name = xsd:anyURI {required}
  functional = xsd:boolean
  inverseFunctional(L,D) = xsd:boolean {fixed as 'false'}
  inverseFunctional(F) = xsd:boolean
  deprecated = xsd:boolean
>
  Content: ((superProperty | domain | range)*)
</DatatypeProperty>
```

Attribute:	**name** - a reference to a name of this property **functional** - asserts that this property can only have one (unique) value for each instance, if true **inverseFunctional** - asserts this property has a range value that uniquely determines a domain value **deprecated** - this property is deprecated if true
Parents:	`Ontology`
Note:	This element contains a value range of data values, and thus link individuals to data values.
See also:	owl:DatatypeProperty owl:FunctionalProperty, owl:InverseFunctionalProperty, and owl:DeprecatedProperty in [OWL Reference]

Example 2_4-1 (L,D,F)

```
<owlx:DatatypeProperty owlx:name="age">
  <owlx:domain owlx:class="Person" />
  <owlx:range owlx:datatype="&xsd;integer" />
</owlx:DatatypeProperty>
```

See also the other examples of datatype property: `timeStamp` in 2.6 and `tennisGameScore` in 2.6.2.

element **ObjectProperty**

```
<ObjectProperty
  name = xsd:anyURI {required}
  inverseOf = xsd:anyURI
  transitive = xsd:boolean
  symmetric = xsd:boolean
  functional = xsd:boolean
  inverseFunctional(L) = xsd:boolean {fixed as 'false'}
  inverseFunctional(D,F) = xsd:boolean
  deprecated = xsd:boolean
>
  Content: ((superProperty | domain | range)*)
</ObjectProperty>
```

Attribute:	**name** - a reference to a name of this property **inverseOf** - a reference to a name of an inverse relation **transitive** - asserts that this property is a transitive relation, if true **symmetric** - asserts that this property is a symmetric relation, if true **functional** - asserts that this property can only have one (unique) value for each instance, if true

inverseFunctional - asserts this property has a range value that uniquely determines a domain value
deprecated - this property is deprecated if true

Parents: `Ontology`

Note: This element contains a value range of class individuals, and thus link individuals to individuals ("individual-valued property" would probably have been a bit better term).

See also: owl:ObjectProperty, owl:inverseOf, owl:TransitiveProperty, owl:SymmetricProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty, and owl:DeprecatedProperty in [OWL Reference]

A property axiom defines characteristics of a properties. In its simplest form, a property axiom just defines the existence of a property. For example:

Example 2_4-2 (L,D,F)

```
<owlx:ObjectProperty owlx:name="hasParent" />
```

This defines a property with the restriction that the range values should be individuals (i.e., `ObjectProperty`).

element **superProperty**

```
<superProperty
  name = xsd:anyURI {required}
>
  Content: (##empty)
</superProperty>
```

Attribute: **name** - a reference to a super property name

Parents: `DatatypeProperty`, `ObjectProperty`

Note: This element can be used for both datatype properties and object properties.

The next example states that all instances (pairs) contained in the property extension of the property `hasMother` are also members of the property extension of the property `hasParent`.

Example 2_4-3 (L,D,F)

```
<owlx:ObjectProperty owlx:name="hasMother">
  <owlx:superProperty owlx:name="#hasParent" />
</owlx:ObjectProperty>
```

element **domain**

```
<domain
  class = xsd:anyURI
>
  Content(L): ( Class[ID]* )
  Content(D,F): ( description* )
</domain>
```

Attribute: **class** - a reference to a class name

Parents: `DatatypeProperty`, `ObjectProperty`

Note: This element asserts that the domain values of this property must belong to the class extension of the class description.

See also: rdfs:domain in [OWL Reference]

For example, it is possible to state that the domain of the property `hasBankAccount` can be either a `Person` or a `Corporation` in the following manner:

Example 2_4-4 (D,F)

```
<owlx:ObjectProperty owlx:name="hasBankAccount">
  <owlx:domain>
    <owlx:UnionOf>
      <owlx:Class owlx:name="#Person" />
      <owlx:Class owlx:name="#Corporation" />
    </owlx:UnionOf>
  </owlx:domain>
</owlx:ObjectProperty>
```

element DatatypeProperty/**range**

```
<range
  datatype = xsd:anyURI
>
  Content(D,F): ( OneOf[data]? )
</range>
```

Attribute:	**datatype** - a reference to a datatype
Parents:	DatatypeProperty
Note:	This element asserts that the range values of this property must belong to data values in the specified data range.
See also:	rdfs:range in [OWL Reference]

element ObjectProperty/**range**

```
<range
  class = xsd:anyURI
>
  Content(L): ( Class[ID]* )
  Content(D,F): ( description* )
</range>
```

Attribute:	**class** - a reference to a class name
Parents:	ObjectProperty
Note:	This element asserts that the range values of this property must belong to the class extension of the class description in the specified data range.
See also:	rdfs:range in [OWL Reference]

element **SubPropertyOf**

```
<SubPropertyOf
  sub = xsd:anyURI {required}
>
  Content: ( DatatypeProperty[ID] | ObjectProperty[ID] )
</SubPropertyOf>
```

Attribute:	**sub** - a reference to a subproperty
Parents:	Ontology
Note:	This element defines that a property specified as a value of `sub` attribute, is a subproperty of another property specified as content.
See also:	owl:subPropertyOf in [OWL Reference]

element **EquivalentProperties**

```
<EquivalentProperties>
  Content: ( (DatatypeProperty[ID], DatatypeProperty[ID]+ ) |
             (ObjectProperty[ID], ObjectProperty[ID]+) )
</EquivalentProperties>
```

Parents:	Ontology
Note:	This element asserts that two or more properties have the same property extension.
See also:	owl:equivalentProperty in [OWL Reference]

element **DatatypeProperty**[ID]

```
<DatatypeProperty
  name = xsd:anyURI {required}
>
  Content: (##empty)
</DatatypeProperty>
```

Attribute:	**name** - a reference to a datatype property
Parents:	SubPropertyOf, EquivalentProperties
Note:	This element is used for solely referring to a datatype property, and does not actually define any property, unlike DatatypeProperty construct.

element **ObjectProperty**[ID]

```
<ObjectProperty
  name = xsd:anyURI {required}
>
  Content: (##empty)
</ObjectProperty>
```

Attribute:	**name** - a reference to an object property
Parents:	SubPropertyOf, EquivalentProperties
Note:	This element is used for solely referring to an object property, and does not actually define any property, unlike ObjectProperty construct.

2.5 Individuals

2.5.1 Individual axioms

Individual axioms (also called "facts") are statements about individuals, indicating class membership and statements about relevant properties.

element **Individual**[axiom]

```
<Individual
  name = xsd:anyURI
>
  Content: (Annotation*, (type | DataPropertyValue | ObjectPropertyValue)* )
</Individual>
```

Attribute:	**name** - a reference to a name of this individual
Parents:	Ontology

Note:	This element indicates class membership and statements about relevant properties.
See also:	owl:Thing in [OWL Reference]

As an example, consider the following set of statements about an instance of the class `Opera` (see also 6.1 in [OWL Reference]):

Example 2_5-1 (L,D,F)

```
<owlx:Individual owlx:name="Tosca">
  <owlx:type owlx:name="Opera" />
  <owlx:ObjectPropertyValue owlx:property="hasComposer">
    <owlx:Individual owlx:name="#Giacomo_Puccini" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hasLibrettist">
    <owlx:Individual owlx:name="#Victorien_Sardou" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hasLibrettist">
    <owlx:Individual owlx:name="#Luigi_Illica" />
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="premiereDate">
    <owlx:DataValue owlx:datatype="&xsd;date">1900-01-14</owlx:DataValue>
  </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="premierePlace">
    <owlx:Individual owlx:name="#Roma" />
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="numberOfActs">
    <owlx:DataValue owlx:datatype="&xsd;positiveInteger">3</owlx:DataValue>
  </owlx:DataPropertyValue>
</owlx:Individual>
```

element **type**

```
<type
  name = xsd:anyURI
>
  Content: ( description* )
</type>
```

Attribute: **name** - a reference to this type

Parents: `Individual`[axiom]

Note: This element specifies type information of a parent individual.

element **DataPropertyValue**

```
<DataPropertyValue
  property = xsd:anyURI {required}
>
  Content: ( DataValue* )
</DataPropertyValue>
```

Attribute: **property** - a reference to a data valued property

Parents: `Individual`[axiom]

Note: This element specifies a data property value to be associated with a parent individual.

element **ObjectPropertyValue**

```
<ObjectPropertyValue
  property = xsd:anyURI {required}
>
```

```
  Content: ( Individual[axiom]* )
</ObjectPropertyValue>
```

Attribute: **property** - a reference to an individual valued property

Parents: `Individual`[axiom]

Note: This element specifies an individual property value to be associated with a parent individual.

Individual axioms need not necessarily be about named individuals: they can also refer to anonymous individuals. As an example, consider the example below. The example defines some facts about an anonymous instance of the class Measurement, a quantitative observation for which facts such as the observed subject, the observed phenomenon, the observed value, and the observation time are listed (see also 6.1 in [OWL Reference]):

Example 2_5-2 (L,D,F)

```
<owlx:Individual>
  <owlx:type owlx:name="Measurement" />
  <owlx:ObjectPropertyValue owlx:property="observedSubject">
    <owlx:Individual owlx:name="#JaneDoe" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="observedPhenomenon">
    <owlx:Individual owlx:name="#Weight" />
  </owlx:ObjectPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="observedValue">
    <owlx:Individual>
      <owlx:type owlx:name="Quantity" />
      <owlx:DataPropertyValue owlx:property="quantityValue">
        <owlx:DataValue
          owlx:datatype="&xsd;float">59.5</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:ObjectPropertyValue owlx:property="quantityUnit">
        <owlx:Individual owlx:name="#Kilogram" />
      </owlx:ObjectPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
  <owlx:DataPropertyValue owlx:property="timeStamp">
    <owlx:DataValue
      owlx:datatype="&xsd;dateTime">2003-01-24T09:00:08+01:00</owlx:DataValue>
  </owlx:DataPropertyValue>
</owlx:Individual>
```

2.5.2 Individual identity

Many languages have a so-called "unique names" assumption: different names refer to different things in the world. On the web, such an assumption is not possible. For example, the same person could be referred to in many different ways (i.e. with different URI references). For this reason OWL does not make this assumption. Unless an explicit statement is being made that two URI references refer to the same or to different individuals, OWL tools should in principle assume either situation is possible.

Two elements `SameIndividual` and `DifferentIndividuals` are provided for making statements about the identity of individuals.

element **SameIndividual**

```
<SameIndividual>
  Content: ( Individual[ID]* )
</SameIndividual>
```

Parents:	Ontology
Note:	This element indicates that two or more URI references actually refer to the same thing: the individuals have the same "identity".
See also:	owl:sameAs and owl:sameIndividualAs in [OWL Reference]

element **Individual**[ID]

```
<Individual
  name = xsd:anyURI {required}
  type = xsd:anyURI
>
  Content: (##empty)
</Individual>
```

Attribute:	**name** - a reference to an individual **type** - a reference to a type of an individual
Parents:	SameIndividual, DifferentIndividuals, EnumeratedClass(D,F), OneOf[object](D,F)
Note:	This element is used for solely referring to an individual ID, and does not actually define any individual, unlike an individual axiom.

For example, we could state that the following two URI references actually refer to the same person (see also 6.2 in [OWL Reference]):

Example 2_5-3 (L,D,F)

```
<owlx:SameIndividual>
  <owlx:Individual owlx:name="#William_Jefferson_Clinton" owlx:type="Human" />
  <owlx:Individual owlx:name="#BillClinton" />
</owlx:SameIndividual>
```

In OWL Full, where class can be treated as instances of (meta)classes, we can use the `SameIndividual` element to define class equality, thus indicating that two concepts have the same intensional meaning. An example (see also 6.2 in [OWL Reference]):

Example 2_5-4 (L,D,F)

```
<owlx:Individual owlx:name="FootballTeam">
  <owlx:type owlx:name="&owlx;Class" />
</owlx:Individual>

<owlx:SameIndividual>
  <owlx:Individual owlx:name="FootballTeam" />
  <owlx:Individual owlx:name="http://sports.org/US#SoccerTeam" />
</owlx:SameIndividual>
```

Note that the first `Individual` element in the above example is an individual axiom, while the other `Individual` elements contained in `SameIndividual` are individual IDs.

element **DifferentIndividuals**

```
<DifferentIndividuals>
  Content: ( Individual[ID]* )
</DifferentIndividuals>
```

Parents:	Ontology
Note:	This element indicates that two or more URI references refer to different individuals.

See also: owl:differntFrom and owl:AllDifferent in [OWL Reference]

The following example states that there are three operas, which are all different individuals (see also 6.2 in [OWL Reference]).

Example 2_5-5 (L,D,F)

```
<owlx:Individual owlx:name="Don_Giovanni">
  <owlx:type owlx:name="Opera" />
</owlx:Individual>

<owlx:Individual owlx:name="Nozze_di_Figaro">
  <owlx:type owlx:name="Opera" />
</owlx:Individual>

<owlx:Individual owlx:name="Cosi_fan_tutte">
  <owlx:type owlx:name="Opera" />
</owlx:Individual>

<owlx:DifferentIndividuals>
  <owlx:Individual owlx:name="#Don_Giovanni" />
  <owlx:Individual owlx:name="#Nozze_di_Figaro" />
  <owlx:Individual owlx:name="#Cosi_fan_tutte" />
</owlx:DifferentIndividuals>
```

2.6 Datatypes

OWL uses the facilities of XML Schema Datatypes [XMLSchema-2], and the following built-in XML Schema datatypes can be used.

xsd:string	xsd:boolean	xsd:decimal
xsd:float	xsd:double	xsd:dateTime
xsd:time	xsd:date	xsd:gYearMonth
xsd:gYear	xsd:gMonthDay	xsd:gDay
xsd:gMonth	xsd:hexBinary	xsd:base64Binary
xsd:anyURI	xsd:normalizedString	xsd:token
xsd:language	xsd:NMTOKEN	xsd:Name
xsd:NCName	xsd:integer	xsd:nonPositiveInteger
xsd:negativeInteger	xsd:long	xsd:int
xsd:short	xsd:byte	xsd:nonNegativeInteger
xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort
xsd:unsignedByte	xsd:positiveInteger	

The specific considerations with the other built-in XML Schema datatypes are explained in the Abstract Syntax document [OWL Semantics].

A datatype can be specified as a value of datatype attribute, which is decleared in allValuesFrom and someValuesFrom elements (for DataRestriction) as well as range element (for DatatypeProperty). A value of the datatype attribute should be a canonical URI reference to an XML Schema datatype: "http://www.w3.org/2001/XMLSchema#*name*", where *name* is a local name of a built-in XML Schema datatype. For example (see also 7.1 in [OWL Reference]):

Example 2_6-1 (L,D,F)

```
<owlx:DatatypeProperty owlx:name="#timeStamp">
  <owlx:domain owlx:class="#Measurement" />
  <owlx:range owlx:datatype="http://www.w3.org/2001/XMLSchema#dateTime" />
</owlx:DatatypeProperty>
```

Note here that XML entities can be used as abbreviations. For example, an XML entity `xsd` is defined in the DOCTYPE declaration:

```
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
```

Then, the above `owlx:datatype` attribute of `owlx:range` element may be specifid in the following manner.

```
<owlx:range owlx:datatype="&xsd;dateTime" />
```

2.6.1 Data value

Data values can be either plain (no datatype) or typed. In DataValue element, typed value may be specified by using `datatype` attribute.

element **DataValue**

```
<DataValue
  datatype = xsd:anyURI
>
  Content: (#CDATA)
</DataValue>
```

Attribute: **datatype** - a reference to a datatype

Parents: `DataPropertyValue`, `OneOf`$^{[data]}_{(D,F)}$

For example, a type of time stamp property value can be specified with the `dateTime` datatype defined as an XML Schema datatype, by using the `datatype` attribute (see also 7.1 in [OWL Reference]).

Example 2_6-2 (L,D,F)

```
<owlx:Individual>
  <owlx:type owlx:name="Measurement" />
  <owlx:DataPropertyValue owlx:property="timeStamp">
    <owlx:DataValue
      owlx:datatype="&xsd;dateTime">2003-01-24T09:00:08+01:00</owlx:DataValue>
  </owlx:DataPropertyValue>
</owlx:Individual>
```

2.6.2 Enumeration of data values

A range of data values can be defined as an enumerated datatype.

element **OneOf** $^{[data]}_{(D,F)}$

```
<OneOf>
  Content: ( DataValue* )
</OneOf>
```

Parents:	`DataRestriction/`allValuesFrom`(D,F)`, `DataRestriction/`someValuesFrom`(D,F)`, `DatatypeProperty/`range`(D,F)`
Note:	This element defines an enumeration of data values. On the other hand, enumeration of individuals is done by `OneOf`[obj].
See also:	Enumerated datatype in [OWL Reference]

For example, the range of `tennisGameScore` property to be the list of integer values {0, 15, 30, 40} can be specifies as below (see also 7.2 in [OWL Reference]):

Example 2_6-3 (D,F)

```
<owlx:DatatypeProperty owlx:name="tennisGameScore">
  <owlx:range>
    <owlx:OneOf>
      <owlx:DataValue owlx:datatype="&xsd;integer">0</owlx:DataValue>
      <owlx:DataValue owlx:datatype="&xsd;integer">15</owlx:DataValue>
      <owlx:DataValue owlx:datatype="&xsd;integer">30</owlx:DataValue>
      <owlx:DataValue owlx:datatype="&xsd;integer">40</owlx:DataValue>
    </owlx:OneOf>
  </owlx:range>
</owlx:DatatypeProperty>
```

3. Element Index and Cross Reference

3.1 Element Index

(Unique parent if any)	**Element name**	**Specified in**
DataRestriction /	allValuesFrom	2.3.2 Property restrictions
ObjectRestriction /	allValuesFrom	2.3.2 Property restrictions
	Annotation	2.2 Header Elements
Ontology /	BackwardCompatibleWith	2.2 Header Elements
	cardinality	2.3.2 Property restrictions
Ontology /	Class[axiom]	2.3 Classes
	Class[ID]	2.3.1 Class descriptions
description /	ComplementOf(D,F)	2.3.4 Boolean Combination
Individual /	DataPropertyValue	2.5.1 Individual axioms
description /	DataRestriction	2.3.2 Property restrictions
Ontology /	DatatypeProperty	2.4 Properties
SubPropertyOf /	DatatypeProperty[ID]	2.4 Properties
	DataValue	2.6.1 Data value
	description	2.3.1 Class descriptions
Ontology /	DifferentIndividuals	2.5.2 Individual identity
Ontology /	DisjointClasses(D,F)	2.3.5 Class relationships
Annotation /	Documentation	2.2 Header Elements
	domain	2.4 Properties

Ontology /	EnumeratedClass(D,F)	2.3.3	Enumeration of individuals
Ontology /	EquivalentClasses	2.3.5	Class relationships
Ontology /	EquivalentProperties	2.4	Properties
DataRestriction /	hasValue(D,F)	2.3.2	Property restrictions
ObjectRestriction /	hasValue(D,F)	2.3.2	Property restrictions
Ontology /	Imports	2.2	Header Elements
Ontology /	IncompatibleWith	2.2	Header Elements
Ontology /	Individual[axiom]	2.5.1	Individual axioms
	Individual[ID]	2.5.2	Individual identity
description /	IntersectionOf(D,F)	2.3.4	Boolean Combination
description /	Label	2.2	Header Elements
	maxCardinality	2.3.2	Property restrictions
	minCardinality	2.3.2	Property restrictions
Ontology /	ObjectProperty	2.4	Properties
SubPropertyOf /	ObjectProperty[ID]	2.4	Properties
Individual /	ObjectPropertyValue	2.5.1	Individual axioms
description /	ObjectRestriction	2.3.2	Property restrictions
	OneOf [data](D,F)	2.6.2	Enumeration of data values
description /	OneOf [object](D,F)	2.3.3	Enumeration of individuals
	Ontology	2.1	The Root Element
Ontology /	PriorVersion	2.2	Header Elements
DatatypeProperty /	range	2.4	Properties
ObjectProperty /	range	2.4	Properties
Ontology /	SameIndividual	2.5.2	Individual identity
DataRestriction /	someValuesFrom	2.3.2	Property restrictions
ObjectRestriction /	someValuesFrom	2.3.2	Property restrictions
SubClassOf /	sub(D,F)	2.3.5	Class relationships
Ontology /	SubClassOf(D,F)	2.3.5	Class relationships
Ontology /	SubPropertyOf	2.4	Properties
SubClassOf /	super(D,F)	2.3.5	Class relationships
	superProperty	2.4	Properties
Individual /	type	2.5.1	Individual axioms
description /	UnionOf(D,F)	2.3.4	Boolean Combination
Ontology /	VersionInfo	2.2	Header Elements

3.2 Cross Reference

The table below shows the term correspondence between RDF/XML and the XML presentation syntax. In the column of presentation syntax, the terms begin with '@' indicate attributes belong to an element concatenated before the attribute with a '/' delimiter.

Note: Appendix A of the OWL Reference [OWL Reference] provides a systematic set of links for each language construct in the Reference document and corresponding sections in the Guide [OWL Guide] as well as the Semantics [OWL Semantics] documents.

RDF/XML in [OWL Reference]	XML Presentation Syntax (this document)
owl:AllDifferent	DifferentIndividuals
owl:allValuesFrom	DataRestriction / allValuesFrom ObjectRestriction / allValuesFrom
owl:backwardCompatibleWith	BackwardCompatibleWith
owl:cardinality	cardinality / @value
owl:Class	Class[axiom]
rdfs:comment	Documentation
owl:complementOf	ComplementOf
rdfs:Datatype	DataRestriction / allValuesFrom / @datatype DataRestriction / someValuesFrom / @datatype DatatypeProperty/ range / @datatype DataValue / @datatype
owl:DatatypeProperty	DatatypeProperty
owl:DeprecatedClass	Class[axiom] / @deprecated EnumeratedClass / @deprecated
owl:DeprecatedProperty	DatatypeProperty / @deprecated ObjectProperty / @deprecated
owl:differentFrom	DifferentIndividuals
owl:disjointWith	DisjointClasses
owl:distinctMembers	
rdfs:domain	domain
owl:equivalentClass	EquivalentClasses
owl:equivalentProperty	EquivalentProperties
owl:FunctionalProperty	DatatypeProperty / @functional ObjectProperty / @functional
owl:hasValue	DataRestriction / hasValue ObjectRestriction / hasValue
owl:imports	Imports
owl:incompatibleWith	IncompatibleWith
owl:intersectionOf	IntersectionOf
owl:InverseFunctionalProperty	DatatypeProperty / @inverseFunctional ObjectProperty / @inverseFunctional
owl:inverseOf	ObjectProperty / @inverseOf
rdfs:label	Label
rdfs:Literal	DataValue

owl:maxCardinality	maxCardinality
owl:minCardinality	minCardinality
owl:Nothing	
owl:ObjectProperty	ObjectProperty
owl:oneOf	EnumeratedClass OneOf [object] OneOf [data]
owl:onProperty	DataRestriction / @property ObjectRestriction / @property
owl:Ontology	
owl:priorVersion	PriorVersion
rdfs:range	DatatypeProperty / range ObjectProperty / range
rdf:RDF	Ontology
owl:Restriction	DataRestriction ObjectRestriction
owl:sameAs	SameIndividual
owl:sameIndividualAs	SameIndividual
owl:someValuesFrom	DataRestriction / someValuesFrom ObjectRestriction / someValuesFrom
rdfs:subClassOf	SubClassOf
rdfs:subPropertyOf	SubPropertyOf
owl:SymmetricProperty	ObjectProperty / @symmetric
owl:Thing	Individual[axiom]
owl:TransitiveProperty	ObjectProperty / @transitive
rdf:type	type
owl:unionOf	UnionOf
owl:versionInfo	VersionInfo