## Primary Components

### Simple Type Definition § 3.14

```
<simpleType
  final = (#all | (list | union | restriction))
  id = ID
  name = NCName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (restriction | list | union))
</simpleType>

<restriction
  base = QName
  id = ID
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (simpleType?, (minExclusive |
           minInclusive | maxExclusive | maxInclusive |
           totalDigits | fractionDigits | length |
           minLength | maxLength | enumeration |
           whiteSpace | pattern)*))
</restriction>

<list
  id = ID
  itemType = QName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (simpleType?))
</list>

<union
  id = ID
  memberTypes = List of QName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (simpleType*))
</union>
```

### Complex Type Definition § 3.4

```
<complexType
  abstract = boolean : false
  block = (#all | List of (extension | restriction))
  final = (#all | List of (extension | restriction))
  id = ID
  mixed = boolean : false
  name = NCName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (simpleContent | complexContent |
           ((group | all | choice | sequence)?,
            ((attribute | attributeGroup)*,
             anyAttribute?))))
</complexType>
```

### Attribute Declaration § 3.2

```
<attribute
  default = string
  fixed = string
  form = (qualified | unqualified)
  id = ID
  name = NCName
  ref = QName
  type = QName
  use = (optional | prohibited | required) : optional
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (simpleType?))
</attribute>
```

### Element Declaration § 3.3

```
<element
  abstract = boolean : false
  block = (#all | List of (extension | restriction |
           substitution))
  default = string
  final = (#all | List of (extension | restriction))
  fixed = string
  form = (qualified | unqualified)
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1
  name = NCName
  nillable = boolean : false
  ref = QName
  substitutionGroup = QName
  type = QName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, ((simpleType | complexType)?,
                          (unique | key | keyref)*))
</element>
```

## Secondary Components

### Attribute Group Definition § 3.6

```
<attributeGroup
  id = ID
  name = NCName
  ref = QName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?,
            ((attribute | attributeGroup)*, anyAttribute?))
</attributeGroup>
```

### Identity Constraint Definition § 3.11

```
<unique
  id = ID
  name = NCName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (selector, field+))
</unique>

<key
  id = ID
  name = NCName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (selector, field+))
</key>

<keyref
  id = ID
  name = NCName
  refer = QName
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (selector, field+))
</keyref>

<selector
  id = ID
  xpath = a subset of XPath expression, see below
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</selector>

<field
  id = ID
  xpath = a subset of XPath expression, see below
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</field>
```

### Model Group Definition § 3.7

```
<group
  name = NCName>
  Content: (annotation?, (all | choice | sequence))
</group>
```

### Notation Declaration § 3.12

```
<notation
  id = ID
  name = NCName
  public = anyURI
  system = anyURI
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</notation>
```

## Helper Components

### Annotations § 3.13

```
<annotation
  id = ID
  {any attributes with non-schema namespace ...}>
  Content: (appinfo | documentation)*
</annotation>

<appinfo
  source = anyURI>
  Content: ({any})*
</appinfo>

<documentation
  source = anyURI
  xml:lang = language>
  Content: ({any})*
</documentation>
```

### Model Groups § 3.8

```
<all
  id = ID
  maxOccurs = 1 : 1
  minOccurs = (0 | 1) : 1
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, element*)
</all>

<choice
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (element | group | choice |
            sequence | any)*)
</choice>

<sequence
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1
  {any attributes with non-schema namespace ...}>
  Content: (annotation?, (element | group | choice |
            sequence | any)*)
</sequence>
```

### Particles § 3.10

Particles correspond to all three elements (<element> not immediately within <schema>, <group> not immediately within <schema> and <any>) which allow minOccurs and maxOccurs attributes. These in turn correspond to two components in each case, a particle and its {term}. The appropriate mapping is described in XML Representation of Element Declaration Schema Components (§3.3.2), XML Representation of Model Group Schema Components (§3.8.2) and XML Representation of Wildcard Schema Components (§3.10.2) respectively.

### Wildcards § 3.10

```
<any
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1
  namespace = ((##any | ##other) | List of (anyURI |
               (##targetNamespace | ##local)) )  : ##any
  processContents = (lax | skip | strict) : strict
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</any>
```

### AttributeUses § 3.11.6

Attribute uses correspond to all uses of <attribute> which allow a use attribute. These in turn correspond to two components in each case, an attribute use and its {attribute declaration} (although note the latter is not new when the attribute use is a reference to a top-level attribute declaration). The appropriate mapping is described in XML Representation of Attribute Declaration Schema Components (§3.2.2).

### Selector Xpath Expressions § 3.11.6

```
Selector::= Path ( '|' Path )*
Path    ::= ('.//')? Step ( '/' Step )*
Step    ::= '.' | NameTest
NameTest::= QName | '*' | NCName ':' '*'
Path    ::= ('.//')? ( Step '/' )* ( Step | '@' NameTest )
```

### Schema Definition § 3.15

```
<schema
  attributeFormDefault =
   (qualified | unqualified) : unqualified
  blockDefault = (#all | List of (extension | restriction
                  | substitution))  : ''
  elementFormDefault =
   (qualified | unqualified) : unqualified
  finalDefault = (#all | List of (extension |
                  restriction))  : ''
  id = ID
  targetNamespace = anyURI
  version = token
  xml:lang = language
  {any attributes with non-schema namespace ...}>
  Content: ((include | import | redefine | annotation)*,
            (((simpleType | complexType | group |
               attributeGroup)
              | element | attribute | notation),
              annotation*)*)
</schema>
```

## Access and Composition

### Include § 4.2.1

```
<include
  id = ID
  schemaLocation = anyURI
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</include>
```

### Redefine § 4.2.2

```
<redefine
  id = ID
  schemaLocation = anyURI
  {any attributes with non-schema namespace ...}>
  Content: (annotation | (simpleType | complexType | group
            | attributeGroup))*
</redefine>
```

### Import § 4.2.3

```
<import
  id = ID
  namespace = anyURI
  schemaLocation = anyURI
  {any attributes with non-schema namespace ...}>
  Content: (annotation?)
</import>
```

# Quick Reference

# XML Schema (XSD)

Part 1: Structures

Version 1.0

W3C Recommendation
02 May 2001

http://www.w3.org/TR/xmlschema-1/

Table of Contents:


**Primary Components**
· Simple Type Definition
· Complex Type Definition
· Attribute Declaration
· Element Declaration
**Secondary Components**
· Attribute Group Definition
· Identity Constraint Definition
· Model Group Definition
· Notation Declaration
**Helper Components**
· Annotations
· Model Groups
· Wildcards
**Miscellaneous**
· Schema Definition
· Selector Xpath Expressions
**Access and Composition**



**deepX Ltd.**

Dublin, Ireland

info@deepX.com
http://www.deepX.com/