XML Schema - Structures Quick Reference

ver 9/01

http://www.xml.dvint.com

1 Namespaces

- http://www.w3.org/2001/XMLSchema
- http://www.w3.org/2001/XMLSchema-instance

2 Schema declaration

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

<include id = ID
schemaLocation = anyURI >
Content: (annotation?) </include>

<redefine id = ID
schemaLocation = anyURI>
Content: (annotation | (simpleType | complexType | group | attributeGroup))*
</redefine>

<import id = ID
namespace = anyURI
schemaLocation = anyURI>
Content: (annotation?) </import>

3 Simple Data Type Declaration

<simpleType id = ID
final = (#all | (list | union | restriction))
name = NCName>
Content: (annotation ?, (restriction | list | union)) </simpleType>

<restriction id = ID
base = QName>
Content: (annotation ?, (simpleType ?, (minExclusive | minInclusive | maxExclusive | maxInclusive | totalDigits | fractionDigits | length | minLength | maxLength | enumeration | whiteSpace | pattern)*)) </restriction>

<list id = ID
itemType = QName>
Content: (annotation ?, (simpleType ?)) </list>

<union id = ID
memberTypes = List of QName>
Content: (annotation ?, (simpleType *)) </union>

Constraining Facets

<length id = ID
fixed = boolean : false
value = nonNegativeInteger >
Content: (annotation?) </length>

<minLength id = ID
fixed = boolean : false
value = nonNegativeInteger >
Content: (annotation?) </minLength>

<maxLength id = ID
fixed = boolean : false
value = nonNegativeInteger >
Content: (annotation?) </maxLength>

<pattern id = ID
value = anySimpleType
Content: (annotation?) </pattern>

<enumeration id = ID
value = anySimpleType >
Content: (annotation?)
</enumeration>

<whiteSpace id = ID
fixed = boolean : false
value = (collapse | preserve | replace)>
Content: (annotation?)
</whiteSpace>

<maxInclusive id = ID
fixed = boolean : false
value = anySimpleType>
Content: (annotation?)
</maxInclusive>

<maxExclusive id = ID
fixed = boolean : false
value = anySimpleType>
Content: (annotation?)
</maxExclusive>

<minExclusive id = ID
fixed = boolean : false
value = anySimpleType>
Content: (annotation?)
</minExclusive>

<minInclusive id = ID
fixed = boolean : false
value = anySimpleType>
Content: (annotation?)
</minInclusive>

<totalDigits id = ID
fixed = boolean : false
value = positiveInteger >
Content: (annotation?)
</totalDigits>

<fractionDigits id = ID
fixed = boolean : false
value = nonNegativeInteger >
Content: (annotation?)
</fractionDigits>

4 Complex Data Type Declaration

<complexType id = ID
abstract = boolean : false
block = (#all | List of (extension | restriction))
final = (#all | List of (extension | restriction))
mixed = boolean : false
name = NCName>
Content: (annotation?, (simpleContent | complexContent | ((group | all | choice | sequence)?, ((attribute | attributeGroup)*, anyAttribute?)))) </complexType>

Complex Content

<complexContent id = ID
mixed = boolean>
Content: (annotation?, (restriction | extension)) </complexContent>

<restriction id = ID
base = QName>
Content: (annotation?, (group | all | choice | sequence)?, ((attribute | attributeGroup)*, anyAttribute?)) </restriction>

<extension id = ID
base = QName>
Content: (annotation?, ((group | all | choice | sequence)?, ((attribute | attributeGroup)*, anyAttribute?))) </extension>

Simple Content

<simpleContent id = ID>
Content: (annotation?, (restriction | extension)) </simpleContent>

<restriction id = ID
base = QName>
Content: (annotation?, (simpleType?, (minExclusive | minInclusive | maxExclusive | maxInclusive | totalDigits | fractionDigits | length | minLength | maxLength | enumeration | whiteSpace | pattern)*)?, ((attribute | attributeGroup)*, anyAttribute?)) </restriction>

<extension id = ID
base = QName>
Content: (annotation?, ((attribute | attributeGroup)*, anyAttribute?)) </extension>

<attributeGroup id = ID
ref = QName>
Content: (annotation?) </attributeGroup>

<anyAttribute id = ID
namespace = ((##any | ##other) | List of (anyURI | (##targetNamespace | ##local))) : ##any
processContents = (lax | skip | strict) : strict >
Content: (annotation?)</anyAttribute>

5 Model Group Definition

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

<all id = ID
maxOccurs = 1 : 1 minOccurs = (0 | 1) : 1>
Content: (annotation?, element*) </all>

6 Attribute Group Definition

```
<attributeGroup  id = ID
  name = NCName
  ref = QName >
  Content: (annotation?,
          ((attribute | attributeGroup)*, anyAttribute?)) </attributeGroup>
```

7 Element Declaration

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

8 Model Group (content model)

```
<choice  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1}>
  Content: (annotation?, (element | group | choice | sequence | any)*)</choice>

<sequence  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1}>
  Content: (annotation?, (element | group | choice | sequence | any)*) </sequence>

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

9 Attribute Declaration

```
<attribute  id = ID
  default = string
  fixed = string
  form = (qualified | unqualified)
  name = NCName
  ref = QName
  type = QName
  use = (optional | prohibited | required) : optional >
  Content: (annotation?, (simpleType?)) </attribute>
```

10 Notation Declaration

```
<notation  id = ID
  name = NCName
  public = anyURI
  system = anyURI }>
  Content: (annotation?) </notation>
```

11 Annotation Components

```
<annotation  id = ID>
  Content: (appinfo | documentation)* </annotation>

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

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

12 Identity-constriant defintions

```
<unique  id = ID
  name = NCName >
  Content: (annotation?, (selector, field+)) </unique>

<key  id = ID
  name = NCName >
  Content: (annotation?, (selector, field+)) </key>

<keyref  id = ID
  name = NCName
  refer = QName >
  Content: (annotation?, (selector, field+)) </keyref>

<selector  id = ID
  xpath = a subset of XPath expression, see below >
  Content: (annotation?) </selector>

<field  id = ID
  xpath = a subset of XPath expression, see below >
  Content: (annotation?) </field>
```

13 Defined Values

{any} Any element not part of Shema namespace.

#all All of the values listed

#atomic A built-in primitive simple type definition

[final]

list A finite-length (possibly empty) sequence of values

union A combination of the of one or more other datatypes.

restriction Values for constraining facets are specified to a subset of those of its base type.

[namespace]

##any Any namespace (default)

##other Any namespace other than target namespace

##targetNamespace Must belong to the target Namespace of schema

##local Any unqualified XML

[processContents]

strict There must be a top-level declaration for the item available, or the item must have an xsi:type, and must be valid.

skip No constraints at all: the item must simply be well-formed.

lax Validate where you can, don't worry when you can't.

[form]

qualified Namespace qualified

unqualified No namespace qualification

[use]

optional Attribute is optional

prohibited Attribute is prohibited

required Attribute is required to have a value

[value]

preserve The value is the normalized value

replace All occurrences of tab, line feed and carriage return are replaced with space.

collapse Contiguous sequences of spaces are collapsed to a single space, and initial and/or final spaces are deleted.

14 Schema Instance Related Markup

xsi:type An element in an instance may explicitly assert its type using the attribute xsi:type. The value is a QName associated with a type definition.

xsi:nil An element may be valid without content if it has the attribute xsi:nil with the value true.

xsi:schemaLocation, xsi:noNamespaceSchemaLocation Provide hints as to the physical location of schema documents.

xmlhelp@dvint.com
http://www.xml.dvint.com