

F

XML Schema Datatypes Reference

This appendix provides a quick reference to the W3C Recommendation for XML Schemas, Part 2: Datatypes. Datatypes were separated into a specification in their own right so that XML Schemas as well as other XML-related technologies (for example, RELAX NG) can use them.

The XML Schema defines several datatypes that can be used to validate the content of attributes and text-only elements. These datatypes enable you to specify that the content must be formatted as a date, a Boolean, a floating-point number, and so on. The second part of the XML Schema Recommendation defines two sorts of datatype:

- ❑ Built-in types, which are available to all XML Schema authors, and should be implemented by a conforming processor
- ❑ User-derived types, which are defined in individual schema documents, and are particular to that schema (although it is possible to import and reuse these definitions in other XML Schemas). These types are based on the existing built-in types.

Built-in types include two subgroups:

- ❑ Built-in primitive types, which are types in their own right. They are not defined in terms of other datatypes. Primitive types are also known as *base types* because they are the basis from which all other types are built.
- ❑ Built-in derived types, which are built from definitions of other primitive and derived datatypes

The first part of this appendix provides a quick overview of all the XML built-in datatypes, both primitive and derived. The second part provides details about all of the constraining *facets*, or characteristics, of these datatypes. Facets can be used to restrict the allowed set of values for a datatype. Also provided in this appendix are tables that illustrate which of these constraining facets can be applied to which datatype.

XML Schema Built-in Datatypes

The following table shows the primitive types that XML Schemas offer, from which you can derive other datatypes. Many of the datatypes have limitations on the maximum or minimum value; others require that the value match a specific format. When an XML Schema validator is used to check instance documents, it determines whether the content matches the declarations in the XML Schema. If a value is declared to be a date and is not formatted correctly, the XML Schema validator will raise a validity error.

Primitive Type	Description	Example
string	Represents any legal character string in XML that matches the Char production in XML 1.0 (http://www.w3.org/TR/REC-xml).	This is a string If you need to include a character that is not easily typed, such as the copyright symbol, or one that may not appear directly in content, you can use a general or character entity reference. Entity references are replaced before validation of content occurs.
boolean	Represents binary logic, true or false	true, false, 1, 0 These are the only permitted values for this datatype.
decimal	Represents a subset of real numbers that can be shown using numerical digits. The decimal point and trailing zeroes are optional.	3.141 The plus sign (+) and minus sign (-) may be used to represent positive or negative numbers — for example, -1.23, +00042.00.
float	Standard concept of real numbers patterned after an IEEE single-precision 32-bit floating-point type. The values INF, -INF, -0, and NaN are permitted.	-INF, -1E4, 4.5E-2, 37, INF, NaN NaN denotes <i>not a number</i> and is neither less than nor greater than any other number. It cannot be compared with other numbers. INF denotes infinity.

Appendix F: XML Schema Datatypes Reference

Primitive Type	Description	Example
double	<p>Standard concept of real numbers patterned after an IEEE double-precision 64-bit floating-point type. The values <code>INF</code>, <code>-INF</code>, <code>-0</code>, and <code>NaN</code> are permitted.</p>	<p><code>-INF</code>, <code>765.4321234E11</code>, <code>7E7</code>, <code>1.0</code>, <code>INF</code>, <code>NaN</code></p> <p><code>NaN</code> denotes <i>not a number</i> and is neither less than nor greater than any other number. It cannot be compared with other numbers.</p> <p><code>INF</code> denotes infinity.</p>
duration	<p>Represents a duration of time in the format <code>PnYnMnDTnHnMnS</code>, where</p> <p><code>P</code> is a designator that must always be present. An optional <code>+</code> or <code>-</code> sign is allowed before <code>P</code>.</p> <p><code>nY</code> represents number of years</p> <p><code>nM</code> represents number of months</p> <p><code>nD</code> represents number of days</p> <p><code>T</code> is the date/time separator. If any time elements are included in the duration, <code>T</code> must be present.</p> <p><code>nH</code> is number of hours</p> <p><code>nM</code> is number of minutes</p> <p><code>nS</code> is number of seconds. Seconds allows a fractional part (arbitrary precision) to appear after a decimal point.</p> <p>Based on ISO 8601.</p>	<p><code>P1Y0M1DT20H25M30.120S</code></p> <p>1 year and 1 day, 20 hours, 25 minutes and 30.120 seconds.</p> <p>Limited forms of this datatype are also allowed. It is not required to include every part of the duration — for example, <code>P120D</code> denotes 120 days.</p>

Table continued on following page

Appendix F: XML Schema Datatypes Reference

Primitive Type	Description	Example
dateTime	<p>A specific instance in time in the following format:</p> <p>CCYY-MM-DDThh:mm:ss where:</p> <p>A leading minus (-) sign is permitted at the beginning of the value to indicate that the year is negative.</p> <p>CC represents the century</p> <p>YY represents the year</p> <p>MM represents the month</p> <p>DD represents the day</p> <p>T is the date/time separator</p> <p>hh represents hours</p> <p>mm represents minutes</p> <p>ss represents seconds. Seconds allows a fractional part (arbitrary precision) to appear after a decimal point.</p> <p>There is also an optional time zone indicator. The time zone must follow this format:</p> <p>-hh:mm</p> <p>A leading + sign or - minus sign followed by the number of hours and minutes indicates the difference between the local time and UTC. A Z may be used to indicate that the time zone is UTC.</p> <p>Based on ISO 8601.</p>	<p>2004-09-13T14:51:26</p> <p>Represents the 13th of September 2004, at 2:51 and 26 seconds in the afternoon.</p> <p>2004-09-13T14:51:26T-05:00</p> <p>2004-09-13T14:51:26Z</p> <p>In addition to the format requirements, the date and time must be valid. For example, 2004-19-01T14:51:26Z would not be valid because there is no 19th month. Likewise, the day portion could never be 32. (Note that the year 0000 is prohibited in XML Schema version 1.0, and each of the fields CC, YY, MM, DD, hh, and mm must be exactly two digits).</p>

Appendix F: XML Schema Datatypes Reference

Primitive Type	Description	Example
time	<p>Represents an instance of time that occurs every day, in the format <code>hh:mm:ss.sss</code>.</p> <p>Fractional seconds can be added to arbitrary precision and there is an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p><code>14:12:30</code></p> <p>Represents 12 minutes and 30 seconds past 2:00 in the afternoon.</p> <p>In addition to the format requirements, the time must be valid. For example, <code>25:51:26</code> would not be valid because there is no 25th hour.</p>
date	<p>Represents a calendar date from the Gregorian calendar (the whole day) in the format <code>CCYY-MM-DD</code>. A leading + sign or - sign is permitted at the beginning of the value to indicate whether the year is positive or negative.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p><code>2004-09-13</code></p> <p>Represents the 13th of September 2004.</p> <p>In addition to the format requirements, the date and time must be valid. For example, <code>2004-19-01</code> would not be valid because there is no 19th month. Likewise, the day portion could never be 32. (Note that the year 0000 is prohibited in XML Schema version 1.0).</p>
gYearMonth	<p>Represents a month in a year in the Gregorian calendar in the format <code>CCYY-MM</code>. A leading minus (-) sign is permitted at the beginning of the value to indicate that the year is negative.</p> <p>A leading + sign or - sign is permitted at the beginning of the value to indicate whether the year is positive or negative.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p><code>2004-09</code></p> <p>Represents September 2004.</p> <p>In addition to the format requirements, the year and month must be valid. For example, <code>2004-19</code> would not be valid because there is no 19th month. (Note that the year 0000 is prohibited in XML Schema version 1.0).</p>

Table continued on following page

Appendix F: XML Schema Datatypes Reference

Primitive Type	Description	Example
<code>gYear</code>	<p>Represents a year in the Gregorian calendar in the format <code>CCYY</code>. A leading + sign or - sign is permitted at the beginning of the value to indicate whether the year is positive or negative.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p>-0001</p> <p>Represents 1 B.C.E. (or 1 B.C.).</p> <p>(Note that the year 0000 is prohibited in XML Schema version 1.0).</p>
<code>gMonthDay</code>	<p>Represents a recurring day of a recurring month in the Gregorian calendar, in the format <code>--MM-DD</code>. No preceding sign (positive or negative) is permitted.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p>--07-12</p> <p>Represents the 12th of July. Ideal for birthdays, anniversaries, holidays, and recurring events.</p>
<code>gDay</code>	<p>Represents a recurring day of a month in the Gregorian calendar, in the format <code>--DD</code>. No preceding sign (positive or negative) is permitted.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p>---16</p> <p>Represents the 16th day of a month. Ideal for monthly occurrences, such as pay day.</p>
<code>gMonth</code>	<p>Represents a recurring month in the Gregorian calendar, in the format <code>--MM</code>. No preceding sign (positive or negative) is permitted.</p> <p>There is also an optional time zone indicator (see <code>dateTime</code>).</p> <p>Based on ISO 8601.</p>	<p>--01</p> <p>Represents January.</p>
<code>hexBinary</code>	<p>Represents hex-encoded arbitrary binary data</p>	<p>0FB7</p>

Appendix F: XML Schema Datatypes Reference

Primitive Type	Description	Example
base64Binary	Represents Base64-encoded arbitrary binary data. The encoding adheres to RFC 2045.	GpM7
anyURI	Represents a Uniform Resource Identifier (URI). The value can be absolute or relative, and may have an optional fragment identifier.	http://www.example.com mailto://info@example.com mySchemafile.xsd
QName	<p>Represents any XML name qualified by a namespace. This includes a local name together with an optional prefix bound to a namespace and separated by a colon.</p> <p>The XML Namespace Recommendation can be found at: http://www.w3.org/TR/REC-xml-names/. Namespaces are discussed in Chapter 3.</p>	contact:FirstName
NOTATION	<p>Represents the NOTATION type from XML 1.0. There must be a corresponding notation declaration within the XML Schema. Only datatypes derived from a NOTATION base type (by specifying a value for enumeration) are allowed to be used as references to notation declarations.</p> <p>Should only be used for attribute values and in XML Schemas without a target namespace.</p>	<pre><xs:notation name="jpeg" system="JPEGViewer.exe" /> <xs:notation name="png" system="PNGViewer.exe" /> <xs:simpleType name= "imageNotation"> <xs:restriction base= "xs:NOTATION" > <xs:enumeration value= "jpeg" /> <xs:enumeration value= "png" /> </xs:restriction> </xs:simpleType></pre>

Appendix F: XML Schema Datatypes Reference

To create new simple datatypes — known as *derived types* — you place further restrictions on an existing built-in type (or another simple type that has been defined). The type on which you place the restrictions is known as the new type's *base type*. Here is a list of the built-in derived types:

Derived type	Description	Example
normalizedString	Represents whitespace-normalized strings. Whitespace-normalized strings do not contain carriage return (#xD), linefeed (#xA) or tab (#x9) characters. Base type: string	Hello World
token	Represents tokenized strings, which do not contain linefeed (#xA), carriage return (#xD), or tab characters (#x9) and contain no leading or trailing spaces, and no internal sequences of two or more spaces. Base type: normalizedString	One Two Three
language	Natural language identifiers, as defined in RFC 3066. Base type: token	en-GB, en-US, fr
NMTOKEN	Represents the NMTOKEN attribute type from XML 1.0. Should only be used on attributes. An NMTOKEN is a “name token” as defined in XML 1.0 Base type: token	small
NMTOKENS	Represents the NMTOKENS attribute type from XML 1.0. Should be used only on attributes. NMTOKENS is a set of NMTOKEN values separated by XML whitespace characters. Base type: A list of items of type NMTOKEN	small medium large
Name	Represents XML Names as defined in XML 1.0. In most cases a colon is allowed, though its use is discouraged. Base type: token	html, sch:assert, Address
NCName	Represents XML “noncolonized” Names (without the prefix and colon), as defined in the Namespaces in XML recommendation. Base type: Name	Address

Appendix F: XML Schema Datatypes Reference

Derived type	Description	Example
ID	Represents the ID attribute type from XML 1.0. Should be used only on attributes. Base type: NCName	<code><address id="Address1" /></code>
IDREF	Represents the IDREF attribute type from XML 1.0. Should be used only on attributes. Base type: NCName	<code><bill sendTo="Address1" /></code>
IDREFS	Represents the IDREFS attribute type from XML 1.0. Should be used only on attributes. IDREFS is a set of IDREF values separated by XML whitespace characters. Base type: A list of items of type IDREF	<code><employee addresses="Address1 Address2" /></code>
ENTITY	Represents the ENTITY attribute type from XML 1.0. Should be used only on attributes. Base type: NCName	Note that the ENTITY has to be declared as an unparsed entity in a DTD.
ENTITIES	Represents the ENTITIES attribute type from XML 1.0. Should be used only on attributes. ENTITIES is a set of ENTITY values separated by an XML whitespace character. Base type: A list of items of type ENTITY	Note that each ENTITY in the list has to be declared as an unparsed entity in a DTD.
integer	Standard mathematical concept of integer numbers, where no fractional value is allowed Base type: decimal	-4, 0, 2, 7
nonPositiveInteger	Standard mathematical concept of a non-positive integer (includes 0) Base type: integer	-4, -1, 0
negativeInteger	Standard mathematical concept of a negative integer (does not include 0) Base type: nonPositiveInteger	-4, -1
long	An integer between -9223372036854775808 and 9223372036854775807 Base type: integer	-23568323, 52883773203895

Table continued on following page

Appendix F: XML Schema Datatypes Reference

Derived type	Description	Example
int	An integer between -2147483648 and 2147483647 Base type: long	-24781982, 24781924
short	An integer between -32768 and 32767 Base type: int	-31353, -43, 345, 31347
byte	An integer between -128 and 127 Base type: short	-127, -42, 0, 54, 125
nonNegativeInteger	Standard mathematical concept of a non-negative integer (includes 0) Base type: integer	0, 1, 42
unsignedLong	A nonNegativeInteger between 0 and 18446744073709551615. Base type: nonNegativeInteger	0, 356, 38753829383
unsignedInt	An unsignedLong between 0 and 4294967295 Base type: unsignedLong	46, 4255774, 2342823723
unsignedShort	An unsignedInt between 0 and 65535 Base type: unsignedInt	78, 64328
unsignedByte	An unsignedShort between 0 and 255 Base type: unsignedShort	0, 46, 247
positiveInteger	Standard mathematical concept of a positive integer (does not include 0) Base type: nonNegativeInteger	1, 24, 345343

Constraining Facets

The constraining facets defined in the XML Schema Datatypes specification are as follows:

- length
- minLength
- maxLength
- pattern

- enumeration
- whitespace
- maxInclusive
- minInclusive
- maxExclusive
- minExclusive
- totalDigits
- fractionDigits

length

The `length` facet enables you to specify the exact length of a datatype. If the datatype is a string, it specifies the number of characters in it. If it's a list, it specifies the number of items in the list. If the base type is `hexBinary` or `base64Binary`, the length is measured in octets. It can only be used inside a `<restriction>` element, and can contain an `<annotation>` element. For more information, see §4.3.1 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="USA_SSN">
  <xs:restriction base="xs:string">
    <xs:length value="11" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
<code>fixed</code>	<code>boolean</code>	If <code>true</code> , any datatypes derived from this type cannot alter the value of <code>length</code> . The default is <code>false</code> .
<code>id</code>	ID	Gives a unique identifier to the type
<code>value</code>	<code>nonNegativeInteger</code>	The actual length of the datatype. You may not use the <code>length</code> facet and the <code>minLength</code> or <code>maxLength</code> facet in the same datatype declaration.

minLength

The `minLength` facet sets the minimum length of a datatype. If the base type is `string`, it sets the minimum number of characters. If it is a list, it sets the minimum number of members. If the base type is `hexBinary` or `base64Binary`, the length is measured in octets. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.2 of the XML Schema Datatypes Recommendation.

Appendix F: XML Schema Datatypes Reference

Example

```
<xs:simpleType name="Password">
  <xs:restriction base="xs:string">
    <xs:minLength value="5" />
    <xs:maxLength value="20" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
fixed	boolean	If true, any datatypes derived from this type cannot alter the value of <code>minLength</code> . The default is false.
id	ID	Gives a unique identifier to the type
value	nonNegativeInteger	Sets the minimum length of the datatype, which must be a non-negative integer. You may not use the <code>length</code> facet and <code>minLength</code> facet in the same datatype declaration.

maxLength

The `maxLength` factor sets the maximum length of a datatype. If the base type is `string`, it sets the maximum number of characters. If it is a list, it sets the maximum number of members. If the base type is `hexBinary` or `base64Binary`, the length is measured in octets. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.3 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="DesiredItems">
  <xs:restriction base="ItemList">
    <xs:minLength value="0" />
    <xs:maxLength value="3" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
fixed	boolean	If true, any datatypes derived from this type cannot alter the value of <code>maxLength</code> . The default is false.
id	ID	Gives a unique identifier to the type
value	nonNegativeInteger	Sets the maximum length of the datatype, which must be a non-negative integer. You may not use the <code>length</code> facet and <code>maxLength</code> facet in the same datatype declaration.

pattern

The `pattern` facet enables you to restrict any simple datatype by specifying a regular expression. It acts on the lexical representation of the type, rather than the value itself. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. If the `pattern` facet is used in a declaration with the base type `list`, the `pattern` applies to the entire list, not each item. For more information, see §4.3.4 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="USA_SSN">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{3}-[0-9]{2}-[0-9]{4}" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
<code>id</code>	ID	Gives a unique identifier to the type
<code>value</code>	string	The value contained within this attribute is any valid regular expression. The regular expression is implicitly anchored to the start (head) and end (tail) of the string.

enumeration

The `enumeration` facet is used to restrict the values allowed within a datatype to a set of specified values. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.5 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="Sizes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="S" />
    <xs:enumeration value="M" />
    <xs:enumeration value="L" />
    <xs:enumeration value="XL" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
<code>id</code>	ID	Gives a unique identifier to the element
<code>value</code>	<code>anySimpleType</code>	One of the values of an enumerated datatype. Multiple enumeration elements are used for the different value options.

whiteSpace

The `whiteSpace` facet dictates what (if any) whitespace transformation is performed upon the datatype content before validation constraints are tested. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.6 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="token">
  <xs:restriction base="xs:normalizedString">
    <xs:whiteSpace value="collapse" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
<code>fixed</code>	boolean	If <code>true</code> , any datatypes derived from this type cannot alter the value of <code>whiteSpace</code> . The default is <code>false</code> .
<code>id</code>	ID	Gives a unique identifier to the type
<code>value</code>	<code>preserve</code> <code>replace</code> <code>collapse</code>	<p><code>preserve</code> means that all whitespace is preserved as it is declared in the element. If <code>replace</code> is used, then all whitespace characters, such as linefeed (<code>#xA</code>), carriage return (<code>#xD</code>), and tab (<code>#x9</code>), are replaced by single whitespace characters (<code>#x20</code>). <code>collapse</code> means all whitespace characters, such as linefeed (<code>#xA</code>), carriage return (<code>#xD</code>), and tab (<code>#x9</code>), are replaced by single whitespace characters (<code>#x20</code>), and then any series of two or more whitespace characters are collapsed into a single whitespace character.</p> <p>Note that a type with its <code>whiteSpace</code> facet set to <code>replace</code> or <code>preserve</code> cannot be derived from one with a value of <code>collapse</code>, and similarly, one with a value of <code>preserve</code> cannot be derived from one with a value of <code>replace</code>.</p>

maxInclusive

The `maxInclusive` facet sets the *inclusive* upper limit of an ordered datatype (number, date type, or ordered list), so the value stated here is therefore the highest value that can be used in this datatype. `maxInclusive` must be equal to or greater than any value of `minInclusive` and greater than the value of `minExclusive`. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.7 of the XML Schema Datatypes Recommendation.

Example

The following example enables you to pick a number between 1 and 10 (the values 1 and 10 are permitted):

```
<xs:simpleType name="PickANumber">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="10" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
fixed	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>maxInclusive</code> . The default is <code>false</code> .
id	ID	Gives a unique identifier to the type
value	anySimpleType	If the base datatype is numerical, then this would be a number; if a date, then this would be a date. The value must be allowable in the base type.

minInclusive

The `minInclusive` facet sets the *inclusive* lower limit of an ordered datatype (number, date type, or ordered list). The value stated here is therefore the lowest value that can be used in this datatype. `minInclusive` must be equal to or less than any value of `maxInclusive` and must be less than the value of `maxExclusive`. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.10 of the XML Schema Datatypes Recommendation.

Example

The following example enables you to pick a number between 1 and 10 (the values 1 and 10 are permitted):

```
<xs:simpleType name="PickANumber">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="10" />
  </xs:restriction>
</xs:simpleType>
```

Appendix F: XML Schema Datatypes Reference

Attributes

Attribute	Value Space	Description
fixed	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>minInclusive</code> . The default is <code>false</code> .
id	ID	Gives a unique identifier to the type
value	<code>anySimpleType</code>	If the base datatype is numerical, then this would be a number; if a date, then a date. The value must be allowable in the base type.

maxExclusive

The `maxExclusive` facet sets the *exclusive* upper limit of an ordered datatype (number, date type, or ordered list). The `maxExclusive` value is therefore one higher than the maximum value that can be used. `maxExclusive` must be greater than or equal to the value of `minInclusive` and greater than the value of `minExclusive`. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.8 of the XML Schema Datatypes Recommendation.

Example

The following example enables you to pick a number between 0 and 11; however, the values 0 and 11 are not permitted:

```
<xs:simpleType name="PickANumber">
  <xs:restriction base="xs:integer">
    <xs:minExclusive value="0" />
    <xs:maxExclusive value="11" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
fixed	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>maxExclusive</code> . The default is <code>false</code> .
id	ID	Gives a unique identifier to the type
value	<code>anySimpleType</code>	If the base datatype is numerical, then this is a number; if a date, then it is a date. The value must be allowable in the base type, or must be equal to the value of the <code>maxExclusive</code> facet in the base type.

minExclusive

The `minExclusive` facet sets the *exclusive* lower limit of an ordered datatype (number, date type, or ordered list). The `minExclusive` value is therefore one lower than the lowest value the data will allow. `minExclusive` must be less than the value of `maxExclusive` and less than or equal to the value of `maxInclusive`. It is always used inside a `<restriction>` element, and it can contain an `<annotation>` element. For more information, see §4.3.9 of the XML Schema Datatypes Recommendation.

Example

The following example enables you to pick a number between 0 and 11; however, the values 0 and 11 are not permitted:

```
<xs:simpleType name="PickANumber">
  <xs:restriction base="xs:integer">
    <xs:minExclusive value="0" />
    <xs:maxExclusive value="11" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
<code>fixed</code>	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>minExclusive</code> . The default is <code>false</code> .
<code>id</code>	ID	Gives a unique identifier to the type
<code>value</code>	<code>anySimpleType</code>	If the base datatype is numerical, then this is a number; if a date, then a date. The value must be allowable in the base type, or must be equal to the value of the <code>minExclusive</code> facet in the base type.

totalDigits

The `totalDigits` facet applies to all datatypes derived from the `decimal` type. The value stated is the *maximum* number of significant digits allowed for the number (the `totalDigits` value must always be a positive integer). Note that leading zeros and trailing zeros after the decimal point are not considered when counting the total number of digits. Because the facet can only be applied to types derived from `decimal`, there are functional limits on the number of digits and the level of precision that can be expressed. The facet applies to only the value and not text representation. For more information, see §4.3.11 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="InterestRatePercent">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="5" />
    <xs:fractionDigits value="3" />
  </xs:restriction>
</xs:simpleType>
```

Appendix F: XML Schema Datatypes Reference

Attributes

Attribute	Value Space	Description
fixed	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>totalDigits</code> . The default is <code>false</code> .
id	ID	Gives a unique identifier to the type
value	positiveInteger	The maximum number of <code>totalDigits</code> allowed for the value

fractionDigits

The `fractionDigits` facet applies to all datatypes derived from the `decimal` type. The value stated is the *maximum* number of digits in the fractional portion of the number (the `fractionDigits` value is always a *non-negative* integer that is less than or equal to the value of `totalDigits`). Note that trailing zeros after the decimal point are not considered when counting the total number of digits. Because the facet can only be applied to types derived from `decimal`, there are functional limits on the number of digits and the level of precision that can be expressed. The facet applies only to the value, not text representation. For more information, see §4.3.12 of the XML Schema Datatypes Recommendation.

Example

```
<xs:simpleType name="InterestRatePercent">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="5" />
    <xs:fractionDigits value="3" />
  </xs:restriction>
</xs:simpleType>
```

Attributes

Attribute	Value Space	Description
fixed	boolean	If <code>true</code> , then any datatypes derived from this type cannot alter the value of <code>fractionDigits</code> . The default is <code>false</code> .
id	ID	Gives a unique identifier to the type
value	nonNegativeInteger	The actual value of the value <code>fractionDigits</code> attribute. This cannot be any larger than the <code>totalDigits</code> value for the current type or base type.

The following table indicates which of these constraining facets may be applied to which built-in datatypes in order to derive new types:

Appendix F: XML Schema Datatypes Reference

Datatypes	length	min Length	max Length	whiteSpace (allowed values)	pattern	enumeration	min Exclusive	max Exclusive	min Inclusive	max Inclusive	total Digits	fraction Digits
String Types												
string	X	X	X	preserve replace collapse	X	X						
anyURI	X	X	X	collapse	X	X						
NOTATION				collapse	X	X						
QName				collapse	X	X						
Binary Encoding Types												
boolean				collapse	X							
hexBinary	X	X	X	collapse	X	X						
base64Binary	X	X	X	collapse	X	X						
Numeric Types												
decimal				collapse	X	X	X	X	X	X	X	X
float				collapse	X	X	X	X	X	X		
double				collapse	X	X	X	X	X	X		

Table continued on following page

Appendix F: XML Schema Datatypes Reference

Datatypes	length	min Length	max Length	whiteSpace (allowed values)	pattern	enumeration	min Exclusive	max Exclusive	min Inclusive	max Inclusive	total Digits	fraction Digits
Date/Time Types												
duration				collapse	X	X	X	X	X	X		
dateTime				collapse	X	X	X	X	X	X		
date				collapse	X	X	X	X	X	X		
time				collapse	X	X	X	X	X	X		
gYear				collapse	X	X	X	X	X	X		
gYearMonth				collapse	X	X	X	X	X	X		
gMonth				collapse	X	X	X	X	X	X		
gMonthDay				collapse	X	X	X	X	X	X		
gDay				collapse	X	X	X	X	X	X		

Appendix F: XML Schema Datatypes Reference

The following table indicates which of these constraining facets may be applied to which derived built-in datatypes in order to derive new types:

Datatypes	length	min Length	max Length	whiteSpace (allowed values)	pattern	enumeration	min Exclusive	max Exclusive	min Inclusive	max Inclusive	total Digits	frac tion Digits
Types Derived from string												
normalized String	X	X	X	preserve replace collapse	X	X						
token	X	X	X	collapse	X	X						
language	X	X	X	collapse	X	X						
Name	X	X	X	collapse	X	X						
NCName	X	X	X	collapse	X	X						
ID	X	X	X	collapse	X	X						
IDREF	X	X	X	collapse	X	X						
IDREFS	X	X	X	collapse	X	X						
NMTOKEN	X	X	X	collapse	X	X						
NMTOKENS	X	X	X	collapse	X	X						
ENTITY	X	X	X	collapse	X	X						
ENTITIES	X	X	X	collapse	X	X						

Table continued on following page

Appendix F: XML Schema Datatypes Reference

Datatypes	length	min Length	max Length	whiteSpace (allowed values)	pattern	enumeration	min Exclusive	max Exclusive	min Inclusive	max Inclusive	total Digits	fraction Digits
Types Derived from decimal												
integer				collapse	X	X	X	X	X	X	X	0
negative Integer				collapse	X	X	X	X	X	X	X	0
positive Integer				collapse	X	X	X	X	X	X	X	0
non Negative Integer				collapse	X	X	X	X	X	X	X	0
non Positive Integer				collapse	X	X	X	X	X	X	X	0
byte				collapse	X	X	X	X	X	X	X	0
short				collapse	X	X	X	X	X	X	X	0
int				collapse	X	X	X	X	X	X	X	0
long				collapse	X	X	X	X	X	X	X	0
unsignedByte				collapse	X	X	X	X	X	X	X	0
unsignedShort				collapse	X	X	X	X	X	X	X	0
unsignedInt				collapse	X	X	X	X	X	X	X	0
unsignedLong				collapse	X	X	X	X	X	X	X	0