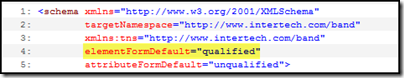
**[XML Schema: elementFormDefault and attributeFormDefault](http://www.intertech.com/Blog/xml-schema-elementformdefault-and-attributeformdefault/)**

* By [Jason Shapiro](http://www.intertech.com/Blog/author/jason-shapiro/)[+](https://plus.google.com/113952051937867131994)
* December 15, 2012
* [Web Services](http://www.intertech.com/Blog/tag/web-services/)
* [0 Comments](http://www.intertech.com/Blog/xml-schema-elementformdefault-and-attributeformdefault/#comments)
* 1 Star2 Stars3 Stars4 Stars5 Stars(No Ratings Yet)

When I teach the XML Schema portion of our [Complete Java Web Services](http://www.intertech.com/Courses/Course?CourseID=99162) class, I eventually reach the subject of **form defaults.**  This is a difficult topic to understand, so I’ve created the following example to simplify the concepts and usage of this attribute.

**The Definition of elementFormDefault**

An XML Schema can define whether its local elements must be qualified or unqualified in an XML document.  This is declared with the **“elementFormDefault”** attribute in the **<schema>** element.   The two possible values are appropriately named: **qualified** and **unqualified.**

[](http://cdn.intertech.com/PostingImages/XML-Schema-Element-Form-Default_8701/efd1.png)

The preceding definition is based on three underlying concepts:

1. Namespaces
2. Qualified vs. Unqualified Elements
3. Global vs. Local Elements

**Namespaces**

In this context, a namespace is a unique string used to disambiguate different XML elements that have been defined with the same name.  Typically the namespaces are defined as URLs, because URLs are more likely to be unique than a random word.  No one outside of Intertech is likely to use **http://www.intertech.com/band** as their namespace, whereas simply using the word **band** might cause a conflict.  A common source of confusion, due to the use of URLs, is the belief that a namespace kicks off some network activity.  The URL here is simply a unique string; there ***may*** be an actual document at the URL specified, but it is not used by the XML parser.

When a namespace is added to an XML document, it is known as a “source namespace” and is defined as the value of an attribute with the following syntax:  **xmlns:<prefix>=”<namespace>”** The prefix used here is prepended to an element associated with that namespace.  Namespaces are available for the element they are defined in and all of their descendants.

For example, here are two “title” elements that need to be treated differently by the parser.  One title is a title of a course and the other title is a job title for an employee.  To disambiguate, two different namespaces are used in this XML document:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <training:class

3: xmlns:training="http://www.intertech.com/training"

4: xmlns:employee="http://www.intertech.com/employee">

5: <training:title>Complete Java Web Services</training:title>

6: <employee:name>Jason Shapiro</employee:name>

7: <employee:title>Instructor</employee:title>

8: </training:class>

In addition, namespaces can be used  to link XML elements to their schema definition.  A schema defines a namespace for all of its elements with the **targetNamespace** attribute.

[](http://cdn.intertech.com/PostingImages/XML-Schema-Element-Form-Default_8701/efd2.png)

When one or more elements in an XML document are to be associated with a schema, their source namespace must match the schema’s targetNamespace. Note that the **band** prefix below has been associated with the same **http://www.intertech.com/band** targetNamespace defined in the schema element.  Now, any element that has **“band:”** as a prefix is associated to that schema.  Also, the **xsi:schemaLocation** attribute is used to associate a namespace with the location of the schema file.

1: <?xml version="1.0" encoding="UTF-8"?>

2: <band:band xmlns:band="http://www.intertech.com/band"

3: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

4: xsi:schemaLocation="http://www.intertech.com/band ./band.xsd">

5: <band:bandName>The Intertech Experience</band:bandName>

6: </band:band>

**Qualified vs. Unqualified Elements**

A qualified element is one that has been associated with a namespace.  This association is made by prepending an element with a prefix defined by an xmlns attribute, as shown above (i.e. in the preceding example, any element that starts with **“band:”** is a qualified element).  It is also possible to specify a qualified namespace without a prefix.  This is known as the **“default namespace”** and its purpose is to associate any elements without a prefix to a specific namespace (thus making them ‘qualified’).  The syntax is: **xmlns=”<namespace>”**

There can only be one namespace associated without a prefix per XML document.

To illustrate this concept, the previous example would be rewritten as:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <!-- The "http://www.intertech.com/band" namespace does not have a prefix.

3: Therefore it is the "default namespace." -->

4: <!-- "band" is qualified in the default namespace -->

5: <band xmlns="http://www.intertech.com/band"

6: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

7: xsi:schemaLocation="http://www.intertech.com/band ./band.xsd">

8: <!-- "bandName" is qualified in the default namespace -->

9: <bandName>The Intertech Experience</bandName>

10: </band>

Once a default namespace has been established for an element, all of its descendants are considered qualified, unless the default namespace is overridden with an empty set of quotes:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <!-- band is qualified in the default namespace -->

3: <band xmlns="http://www.intertech.com/band"

4: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

5: xsi:schemaLocation="http://www.intertech.com/band ./band.xsd">

6: <!-- bandName is qualified in the default namespace -->

7: <bandName>The Intertech Experience</bandName>

8: <!-- yearFormed is unqualified due to the overridden namespace -->

9: <yearFormed xmlns="">1991</yearFormed>

10: </band>

**Global vs. Local Elements**

An element is global or local depending on how it is defined in a schema.  Any element that is an immediate child of Schema, or is a **ref** to an immediate child of Schema, is a **global** element.  All other elements are **local** elements.

**band.xsd**

1: <?xml version="1.0" encoding="UTF-8"?>

2: <schema xmlns="http://www.w3.org/2001/XMLSchema"

3: targetNamespace="http://www.intertech.com/band"

4: xmlns:tns="http://www.intertech.com/band"

5: elementFormDefault="qualified"

6: attributeFormDefault="unqualified">

7:

8: <complexType name="Band\_Member\_Type">

9: <sequence>

10: <!-- LOCAL: not an immediate child of "schema" -->

11: <element name="fullName" type="string" />

12:

13: <!-- GLOBAL: a ref to an immediate child of "schema" -->

14: <element ref="tns:instrument" maxOccurs="unbounded" />

15: </sequence>

16: </complexType>

17:

18: <!-- GLOBAL: an immediate child of "schema" -->

19: <element name="instrument" type="string" />

20:

21: <!-- GLOBAL: an immediate child of "schema" -->

22: <element name="bandName" type="string" />

23:

24: <!-- GLOBAL: an immediate child of "schema" -->

25: <element name="band">

26: <complexType>

27: <sequence>

28: <!-- GLOBAL: a ref to an immediate child of "schema" -->

29: <element ref="tns:bandName" />

30:

31: <!-- LOCAL: not an immediate child of "schema" -->

32: <element name="member" type="tns:Band\_Member\_Type" minOccurs="1"

33: maxOccurs="unbounded" />

34: </sequence>

35: <attribute name="id" type="string" />

36: </complexType>

37: </element>

38: </schema>

**elementFormDefault=”qualified”**

Setting this attribute to **“qualified”** means all **global** and **local** elements from this schema **must** be qualified when used in an XML document.  For example, lets use the band.xsd schema from above, along with the following:

**cd.xsd**

1: <?xml version="1.0" encoding="UTF-8"?>

2: <schema xmlns="http://www.w3.org/2001/XMLSchema"

3: targetNamespace="http://www.intertech.com/cd"

4: xmlns:tns="http://www.intertech.com/cd"

5: xmlns:band="http://www.intertech.com/band"

6: elementFormDefault="qualified"

7: attributeFormDefault="unqualified">

8: <import namespace="http://www.intertech.com/band"

9: schemaLocation="./band.xsd" />

10:

11: <!-- GLOBAL: an immediate child of schema -->

12: <element name="cd">

13: <complexType>

14: <sequence>

15: <!-- GLOBAL: a ref of an immediate child of schema -->

16: <element ref="band:band" />

17:

18: <!-- LOCAL: not an immediate child of schema -->

19: <element name="title" type="string" />

20: </sequence>

21: </complexType>

22: </element>

23: </schema>

Now lets create an XMLdocument from these schemata.  Note that since each schema defines elementFormDefault as qualified, each element must be associated with a namespace.

1: <?xml version="1.0" encoding="UTF-8"?>

2: <cd:cd

3: xmlns:cd="http://www.intertech.com/cd"

4: xmlns:band="http://www.intertech.com/band"

5: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

6: xsi:schemaLocation="http://www.intertech.com/cd ./cd.xsd

7: http://www.intertech.com/band ./band.xsd">

8: <band:band>

9: <band:bandName>The Intertech Experience</band:bandName>

10: <band:member>

11: <band:fullName>Jason Shapiro</band:fullName>

12: <band:instrument>Bass</band:instrument>

13: </band:member>

14: <band:member>

15: <band:fullName>Jim White</band:fullName>

16: <band:instrument>Vocals</band:instrument>

17: </band:member>

18: <band:member>

19: <band:fullName>Davin Mickelson</band:fullName>

20: <band:instrument>Guitar</band:instrument>

21: </band:member>

22: <band:member>

23: <band:fullName>Andrew Trolesen</band:fullName>

24: <band:instrument>Drums</band:instrument>

25: </band:member>

26: </band:band>

27: <cd:title>One</cd:title>

28: </cd:cd>

We could also define one of the namespaces as the “default namespace.”  In this example, we’ll set the **http://www.intertech.com/band** namespace as the default and remove the prefix.  Even without the prefix, all of the elements in this XML document are qualified:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <cd:cd

3: xmlns:cd="http://www.intertech.com/cd"

4: xmlns="http://www.intertech.com/band"

5: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

6: xsi:schemaLocation="http://www.intertech.com/cd ./cd.xsd

7: http://www.intertech.com/band ./band.xsd">

8: <band>

9: <bandName>The Intertech Experience</bandName>

10: <member>

11: <fullName>Jason Shapiro</fullName>

12: <instrument>Bass</instrument>

13: </member>

14: <member>

15: <fullName>Jim White</fullName>

16: <instrument>Vocals</instrument>

17: </member>

18: <member>

19: <fullName>Davin Mickelson</fullName>

20: <instrument>Guitar</instrument>

21: </member>

22: <member>

23: <fullName>Andrew Trolesen</fullName>

24: <instrument>Drums</instrument>

25: </member>

26: </band>

27: <cd:title>One</cd:title>

28: </cd:cd>

**elementFormDefault=”unqualified”**

Setting this attribute to **“unqualified”** means all **global** elements from this schema **must** be **qualified**, and all **local** elements **must** be **unqualified**, when used in an XML document.

Using the previous schemata, with the exception of changing each **elementFormDefault** to **unqualified**, we’re left with a rather odd looking XML document due to a seemingly random collection of elements that require prefixes while others do not.  In this case, **cd**, **band**, **bandName,** and **instrument** require a prefix because they are global elements, and all global elements must be qualified with a namespace.  The rest are local so they do not have a prefix.

1: <?xml version="1.0" encoding="UTF-8"?>

2: <cd:cd

3: xmlns:cd="http://www.intertech.com/cd"

4: xmlns:band="http://www.intertech.com/band"

5: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

6: xsi:schemaLocation="http://www.intertech.com/cd ./cd.xsd

7: http://www.intertech.com/band ./band.xsd">

8: <band:band>

9: <band:bandName>The Intertech Experience</band:bandName>

10: <member>

11: <fullName>Jason Shapiro</fullName>

12: <band:instrument>Bass</band:instrument>

13: </member>

14: <member>

15: <fullName>Jim White</fullName>

16: <band:instrument>Vocals</band:instrument>

17: </member>

18: <member>

19: <fullName>Davin Mickelson</fullName>

20: <band:instrument>Guitar</band:instrument>

21: </member>

22: <member>

23: <fullName>Andrew Trolesen</fullName>

24: <band:instrument>Drums</band:instrument>

25: </member>

26: </band:band>

27: <title>One</title>

28: </cd:cd>

A “default namespace” is illegal when the elementFormDefault is set to unqualified.  This is because the default namespace qualifies all prefix-less elements, and an element cannot be both qualified and unqualified at the same time.

1: <?xml version="1.0" encoding="UTF-8"?>

2:

3: <!-- In this doc all elements without a prefix

4: are qualified to the default namespace -->

5:

6: <!-- LEGAL: global element "cd" is qualified -->

7: <cd:cd

8: xmlns:cd="http://www.intertech.com/cd"

9: xmlns="http://www.intertech.com/band"

10: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

11: xsi:schemaLocation="http://www.intertech.com/cd ./cd.xsd

12: http://www.intertech.com/band ./band.xsd">

13: <!-- LEGAL: global element "band" is qualified -->

14: <band>

15: <!-- LEGAL: global element "bandName" is qualified -->

16: <bandName>The Intertech Experience</bandName>

17: <!-- ILLEGAL: local element "member" cannot be qualified -->

18: <member>

19: <!-- ILLEGAL: local element "fullName" cannot be qualified -->

20: <fullName>Jason Shapiro</fullName>

21: <!-- LEGAL: global element "instrument" is qualified -->

22: <instrument>Bass</instrument>

23: </member>

24: <!-- the rest of the band... -->

25: </band>

26: <!-- ILLEGAL: local element "title" cannot be qualified -->

27: <title>One</title>

28: </cd:cd>

So why would anyone use it?  If we were following patterns that produce a minimal amount of global elements, we could reduce the amount of prefix “bulk” we have in our XML documents by using “**unqualified**.”  For example, lets rewrite our schemata as follows:

**band.xsd**

1: <?xml version="1.0" encoding="UTF-8"?>

2: <schema xmlns="http://www.w3.org/2001/XMLSchema"

3: targetNamespace="http://www.intertech.com/band"

4: xmlns:tns="http://www.intertech.com/band"

5: elementFormDefault="unqualified"

6: attributeFormDefault="unqualified">

7:

8: <complexType name="Band\_Member\_Type">

9: <sequence>

10: <!-- LOCAL: not an immediate child of "schema" -->

11: <element name="fullName" type="string" />

12:

13: <!-- LOCAL: not an immediate child of "schema" -->

14: <element name="instrument" type="string"

15: maxOccurs="unbounded" />

16: </sequence>

17: </complexType>

18: <complexType name="Band\_Type">

19: <sequence>

20: <!-- LOCAL: not an immediate child of "schema" -->

21: <element name="bandName" type="string" />

22:

23: <!-- LOCAL: not an immediate child of "schema" -->

24: <element name="member" type="tns:Band\_Member\_Type" minOccurs="1"

25: maxOccurs="unbounded" />

26: </sequence>

27: </complexType>

28: </schema>

**cd.xsd**

1: <?xml version="1.0" encoding="UTF-8"?>

2: <schema xmlns="http://www.w3.org/2001/XMLSchema"

3: targetNamespace="http://www.intertech.com/cd"

4: xmlns:tns="http://www.intertech.com/cd"

5: xmlns:band="http://www.intertech.com/band"

6: elementFormDefault="unqualified"

7: attributeFormDefault="unqualified">

8: <import namespace="http://www.intertech.com/band"

9: schemaLocation="./band.xsd" />

10:

11: <!-- GLOBAL: an immediate child of schema -->

12: <element name="cd">

13: <complexType>

14: <sequence>

15: <!-- LOCAL: not an immediate child of schema -->

16: <element name="band" type="band:Band\_Type" />

17:

18: <!-- LOCAL: not an immediate child of schema -->

19: <element name="title" type="string" />

20: </sequence>

21: </complexType>

22: </element>

23: </schema>

With so many local elements in our schemata, we are now able to create an XML document with a minimal amount of prefixes:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <cd:cd

3: xmlns:cd="http://www.intertech.com/cd"

4: xmlns:band="http://www.intertech.com/band"

5: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

6: xsi:schemaLocation="http://www.intertech.com/cd ./cd.xsd

7: http://www.intertech.com/band ./band.xsd">

8: <band>

9: <bandName>The Intertech Experience</bandName>

10: <member>

11: <fullName>Jason Shapiro</fullName>

12: <instrument>Bass</instrument>

13: </member>

14: <member>

15: <fullName>Jim White</fullName>

16: <instrument>Vocals</instrument>

17: </member>

18: <member>

19: <fullName>Davin Mickelson</fullName>

20: <instrument>Guitar</instrument>

21: </member>

22: <member>

23: <fullName>Andrew Trolesen</fullName>

24: <instrument>Drums</instrument>

25: </member>

26: </band>

27: <title>One</title>

28: </cd:cd>

**What about ‘attributeFormDefault?’**

This attribute is a bit simpler to understand.  If the value is **“qualified,”** all attributes must be qualified; if it’s **“unqualified,”** attributes must be unqualified.  That’s it!  You do not need to be concerned whether the attribute is global or local.

1: <!-- Attribute definition in schema... -->

2: <attribute name="id" type="string" />

3:

4: <!-- Snippets from inside the XML doc -->

5: <!-- elementFormDefault="qualified" attributeFormDefault="qualified" -->

6: <band:band band:id="Inter001" xmlns:band="http://www.intertech.com/band">

7:

8: <!-- elementFormDefault="qualified" attributeFormDefault="unqualified" -->

9: <band:band id="Inter001" xmlns:band="http://www.intertech.com/band">

Whereas the default namespace causes problems for elements if **elementFormDefault** is set to **unqualified**, attributes have the opposite problem.  Any attribute that does not have a prefix is considered unqualified, even if a default namespace exists.  Therefore if a default namespace has been created for the attribute, and the **attributeFormDefault** is set to **qualified**, there will be a conflict because an attribute cannot be qualified and unqualified at the same time.

1: <!-- elementFormDefault="qualified" attributeFormDefault="qualified" -->

2: <!-- ILLEGAL: id is unqualified (since attributes cannot belong to

3: the default namespace), and the schema requires the attribute

4: to be qualified. -->

5: <band id="Inter001" xmlns="http://www.intertech.com/band">

**Recommended Usage**

In general, because of the potential confusion and odd syntax that can arise from a mix of global and local elements in a single document, along with issues that can arise with the default namespace, it is advised to stick with **elementFormDefault=”qualified”.** It may be more verbose, but it is also much clearer which namespace/schema owns a given element.  The XML document author can always rely on the “default namespace” if they want to reduce the amount of prefixes.  In addition, **attributeFormDefault** should be **unqualified** due to the potential conflict when using a default namespace.

[**XML Schema qualified, unqualified what’s it all about?**](http://michielv.eu/2012/11/12/xml-schema-qualified-unqualified-whats-it-all-about/)

Last modified: 23/11/2012

Nov 12

admin[XML Schema](http://michielv.eu/category/xml-schema/" \o "View all posts in XML Schema) [Eclipse](http://michielv.eu/tag/eclipse/), [XML](http://michielv.eu/tag/xml/) [No Comments](http://michielv.eu/2012/11/12/xml-schema-qualified-unqualified-whats-it-all-about/#respond)

Did you wonder or even struggle with these attributes elementFormDefault="qualified" and attributeFormDefault="unqualified" on the root element of an XML Schema?  
Maybe you even ignored them without even paying attention on them, but still curious?  
Well if you did, take a deep breath, this is the place to be and read on.

**What is an XML namespace?**

“XML namespaces are used for providing uniquely named elements and attributes in an XML document. They are defined in a W3C recommendation. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, the ambiguity between identically named elements or attributes can be resolved.”  
See [Wikipedia](http://en.wikipedia.org/wiki/XML_namespace) for more detailed info.

You can find a good [XML namespace tutorial](http://zvon.org/comp/r/tut-Namespace.html) with nice examples on zvon.org.

**A Schema without target namespace**

Let’s first start with a schema without a target namespace. The *target namespace* is the place were your declared elements and attributes in your schema belong to.  
So in this first example, the elements (A, B, C, D) and attributes (E, F) are not part of any namespace.  
The elements (schema, element, complexType, sequence, …) of the schema itself belong to the namespace http://www.w3.org/2001/XMLSchema.  
The prefix xs is used to refer to this namespace, but you can use whatever prefix you want as long as your namespace declaration has the same prefix.

schema without target namespace

1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2. <!– atts elementFormDefault and attributeFormDefault are irrelvant
3. when no @targettNamespace is used –>
4. <xs:element name="A">
5. <xs:complexType>
6. <xs:sequence>
7. <xs:element name="B" type="xs:string"/>
8. <xs:element name="C" type="xs:string"/>
9. <xs:element ref="D"/>
10. </xs:sequence>
11. <xs:attribute name="E" type="xs:string"/>
12. <xs:attribute ref="F"/>
13. </xs:complexType>
14. </xs:element>
15. <xs:element name="D" type="xs:string"/>
16. <xs:attribute name="F" type="xs:string"/>
17. </xs:schema>

A corresponding instance without a target namespace

1. <?xml version="1.0" encoding="UTF-8"?>
2. <A xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:noNamespaceSchemaLocation="mySchema.xsd"
4. E="valueforAttE" F="valueforAttF">
5. <B/>
6. <C/>
7. <D/>
8. </A>

The attribute xsi:noNamespaceSchemaLocation="mySchema.xsd" is in the http://www.w3.org/2001/XMLSchema-instance namespace and is used to refer to the schema. So this attribute is not part of your schema because it lives in its own namespace.

In this context it has no sense to use elementFormDefault="qualified" or elementFormDefault="unqualified" or attributeFormDefault="qualified" or attributeFormDefault="unqualified".  
It will not harm, you can try all the 4 combinations, it will make no difference, if there is no target namespace.

**The scope of @elementFormDefault and @attributeFormDefault in XML Schema**

So these 2 attributes are only relevant if a **target namespace** is used AND ONLY for **LOCAL declared attributes** and **LOCAL declared elements**.  
If a target namespace is used, GLOBAL declared attributes and GLOBAL declared values MUST be qualified.  
Qualified means: “belongs to a namespace”.  
Unqualified means: “not belonging to a namespace”.  
Why MUST global declared elements and attributes always belong to a namespace if a target namespace is used (and so the choice qualified/unqualified is irrelevant)?  
Because global means you can refer to them from an other namespace and so you need to mention to which namespace you refer.  
If the combination of namespaces with different schemas is not clear, don’t worry I will post a seperate post on this matter.

**@elementFormDefault=”qualified” and @attributeFormDefault=”unqualified”**

1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:foo="http://www.foo.org/"
2. targetNamespace="http://www.foo.org/" elementFormDefault="qualified"
3. attributeFormDefault="unqualified">
4. <xs:element name="A">
5. <xs:complexType>
6. <xs:sequence>
7. <xs:element name="B" type="xs:string"/>
8. <xs:element name="C" type="xs:string"/>
9. <xs:element ref="foo:D"/>
10. </xs:sequence>
11. <xs:attribute name="E" type="xs:string"/>
12. <xs:attribute ref="foo:F"/>
13. </xs:complexType>
14. </xs:element>
15. <xs:element name="D" type="xs:string"/>
16. <xs:attribute name="F" type="xs:string"/>
17. </xs:schema>

The elements A and D and the attribute F are global declared (child elements of the root element <schema>).  
So if we refer to D and F in the schema we must use the namespace (by using its prefix foo).

Example 1

1. <?xml version="1.0" encoding="UTF-8"?>
2. <A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. E="valueforAttE" foo:F="valueforAttF">
6. <B/>
7. <C/>
8. <D/>
9. </A>

The element names A, B, C, D reside in the http://www.foo.org/ namespace. They don’t need a prefix foo (but in this case giving them one is no problem, see example 2 and 3) because the default namespace, declared by xmlns is also http://www.foo.org/.  
The E attribute is local declared and unqualified in the schema so in the document instance it has no namespace.  
The F attribute is global declared and there is a target namespace http://www.foo.org/ in the schema, so in the document instance this attribute must have the prefix referring to that namespace. Attributes without a prefix don’t belong to any namespace, not even the default namespace.

Example 2 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. E="valueforAttE" foo:F="valueforAttF">
6. <!– removing the default namespace on the root element A will give an error –>
7. <B/>
8. <C/>
9. <D/>
10. </foo:A>

Example 3 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
4. E="valueforAttE" foo:F="valueforAttF">
5. <foo:B/>
6. <foo:C/>
7. <foo:D/>
8. </foo:A>

**@elementFormDefault=”qualified” and @attributeFormDefault=”qualified”**

This means all local declared elements and attributes must belong to the target namespace.

1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:foo="http://www.foo.org/"
2. targetNamespace="http://www.foo.org/" elementFormDefault="qualified"
3. attributeFormDefault="qualified">
4. <xs:element name="A">
5. <xs:complexType>
6. <xs:sequence>
7. <xs:element name="B" type="xs:string"/>
8. <xs:element name="C" type="xs:string"/>
9. <xs:element ref="foo:D"/>
10. </xs:sequence>
11. <xs:attribute name="E" type="xs:string"/>
12. <xs:attribute ref="foo:F"/>
13. </xs:complexType>
14. </xs:element>
15. <xs:element name="D" type="xs:string"/>
16. <xs:attribute name="F" type="xs:string"/>
17. </xs:schema>

Example 1

1. <?xml version="1.0" encoding="UTF-8"?>
2. <A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. foo:E="valueforAttE" foo:F="valueforAttF">
6. <B/>
7. <C/>
8. <D/>
9. </A>

Example 2 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. foo:E="valueforAttE" foo:F="valueforAttF">
6. <!– removing the default namespace on the root element A will give an error –>
7. <B/>
8. <C/>
9. <D/>
10. </foo:A>

Example 3 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
4. foo:E="valueforAttE" foo:F="valueforAttF">
5. <foo:B/>
6. <foo:C/>
7. <foo:D/>
8. </foo:A>

**@elementFormDefault=”unqualified” and @attributeFormDefault=”unqualified”**

This means all local declared elements and attributes must not belong to a namespace.

1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:foo="http://www.foo.org/"
2. targetNamespace="http://www.foo.org/"
3. elementFormDefault="unqualified" attributeFormDefault="unqualified">
4. <xs:element name="A">
5. <xs:complexType>
6. <xs:sequence>
7. <xs:element name="B" type="xs:string"/>
8. <xs:element name="C" type="xs:string"/>
9. <xs:element ref="foo:D"/>
10. </xs:sequence>
11. <xs:attribute name="E" type="xs:string"/>
12. <xs:attribute ref="foo:F"/>
13. </xs:complexType>
14. </xs:element>
15. <xs:element name="D" type="xs:string"/>
16. <xs:attribute name="F" type="xs:string"/>
17. </xs:schema>

Element B and C are local declared in the schema so they don’t belong to any namespace hence xmlns="", otherwise they would be in the scope of the default namespace declared on the A element.

Example 1

1. <?xml version="1.0" encoding="UTF-8"?>
2. <A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. E="valueforAttE" foo:F="valueforAttF">
6. <B xmlns=""/>
7. <C xmlns=""/>
8. <D/>
9. </A>

Example 2 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. E="valueforAttE" foo:F="valueforAttF">
6. <B xmlns=""/>
7. <C xmlns=""/>
8. <D/>
9. </foo:A>

Example 3 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. E="valueforAttE" foo:F="valueforAttF">
6. <B/>
7. <C/>
8. <foo:D/>
9. </foo:A>

In this third example, we don’t need to “unnamespace” element B and C because there is no default namespace in scope.

**@elementFormDefault=”unqualified” and @attributeFormDefault=”qualified”**

1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
2. xmlns:foo="http://www.foo.org/" targetNamespace="http://www.foo.org/"
3. elementFormDefault="unqualified" attributeFormDefault="qualified">
4. <xs:element name="A">
5. <xs:complexType>
6. <xs:sequence>
7. <xs:element name="B" type="xs:string"/>
8. <xs:element name="C" type="xs:string"/>
9. <xs:element ref="foo:D"/>
10. </xs:sequence>
11. <xs:attribute name="E" type="xs:string"/>
12. <xs:attribute ref="foo:F"/>
13. </xs:complexType>
14. </xs:element>
15. <xs:element name="D" type="xs:string"/>
16. <xs:attribute name="F" type="xs:string"/>
17. </xs:schema>

Example 1

1. <?xml version="1.0" encoding="UTF-8"?>
2. <A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. foo:E="valueforAttE" foo:F="valueforAttF">
6. <B xmlns=""/>
7. <C xmlns=""/>
8. <D/>
9. </A>

Example 2 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/" xmlns="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. foo:E="valueforAttE" foo:F="valueforAttF">
6. <B xmlns=""/>
7. <C xmlns=""/>
8. <D/>
9. </foo:A>

Example 3 with same schema

1. <?xml version="1.0" encoding="UTF-8"?>
2. <foo:A xmlns:foo="http://www.foo.org/"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.foo.org/ mySchema.xsd"
5. foo:E="valueforAttE" foo:F="valueforAttF">
6. <B/>
7. <C/>
8. <foo:D/>
9. </foo:A>

**Conclusion**

Well this is it, if you read until here, I guess you learned something new.  
All examples were tested and validated with [Eclipse](http://www.eclipse.org/).