

Schemas, BizTalk, and eCommerce

What this chapter covers:

- eCommerce is here to stay and it affects everything!
- Where does Biztalk fit in?
- BizTalk Server and BizTalk Framework
- Why are Schemas so important?
- Getting to know the BizTalk Framework Schemas

8.1 *An introduction to eCommerce*

Looking back on the (sometimes turbulent) history of computing, it is clear that the Internet and specifically the World Wide Web have been an incredible success. They have already had a huge impact on many businesses and markets. In some cases, completely new industries and markets have popped up. That is the power of this new interconnected economy, which has only just begun.

Let's travel back in time and take a look at the growth of the Internet and pick out some key trends, which will demonstrate the imminent impact of eCommerce and the Internet on your business and your relationships with your partners, customers, and competitors.

Table 8.1 shows the total number of registered Internet DNS host names (such as <http://www.news.com>) at different times in the past.

Table 8.1 The growth of DNS hosts on the Internet

Date:	Number of DNS hosts:
1988	30,000
January 1993	1,313,000
July 1993	1,776,000
January 1994	2,217,000
July 1994	3,212,000
January 1995	5,846,000
July 1995	8,200,000
January 1996	14,352,000
July 1996	16,729,000
January 1997	21,819,000
July 1997	26,053,000
January 1998	29,670,000
July 1998	36,739,000
January 1999	43,230,000
July 1999	56,218,000

The first trend that we can assume was instrumental in the massive growth of the Internet was the combination of HTML (which is easy to learn and use) with the inexpensive and wide availability of Internet access. The statistics also show the demand for information and for connectivity between businesses.

With all this activity, the second trend we now see is that more companies are turning to the Internet to buy goods and services from their suppliers and trading partners, placing orders on everything from office supplies to safety equipment to temporary per-

sonnel. IDC research estimates that the Internet commerce procurement market, worth 147 million U.S. dollars in 1998, will be worth five billion U.S. dollars by 2003, with 46% of this activity taking place outside the U.S. Overall, Internet commerce will top one trillion U.S. dollars in the same year.

The third trend we already see is that HTML is becoming simply a part of XML. This flexibility and extensibility has resulted in Schemas, which are themselves XML-based.

8.2 *Why do our systems need a Schema?*

When people talk about eCommerce, they don't mean put a button on the website that says "call 1-800...." No, what they are talking about are computer systems that describe themselves to other machines and computing devices. These systems are able to sustain automated relationships with each other—moving information around, fulfilling just-in-time orders, providing receipts, and more.

Schemas are the key link in that chain, because they provide the reliable structure for data, which is so desperately needed on the Internet. Schemas are therefore, without doubt, the key to providing automated eCommerce between companies.

There are many more reasons to use XML and Schemas.

8.2.1 *Development of flexible web applications*

Because a Schema is basically a vocabulary you have defined to describe your data, you can use the same vocabularies in different programs. You can also query other systems to discover their Schemas and learn to exchange data with that system. If the Schema is stored within a document repository such as BizTalk (see <http://www.biztalk.org>), then entire vertical industries could rely on a select number of Schemas, and the benefits of interoperability would be huge.

Your software development teams will begin programming in a more flexible way, using Schemas and DOM objects to understand data, rather than accessing table structures directly. The benefit of this is substantial. Provided the Schema is maintained and is correct, table designs can change, data can be moved, and new data can be added without breaking your programs.

8.2.2 *So, how does it work then?*

XML is meant to be readable to humans and not overly complex. Here is a very simple XML file for a resumé:

```
<?xml version="1.0" ?>
<PEOPLE xmlns="x-schema:PEOPLE.DTD">
<PERSON>
  <NAME>Mark Wilson</NAME>
```

```

<ADDRESS>911 Somewhere Circle, Canberra, Australia</ADDRESS>
<TEL>(++612) 12345</TEL>
<FAX>(++612) 12345</FAX>
<EMAIL>Mark.Wilson@somewhere.com</EMAIL>
</PERSON>
</PEOPLE>

<PEOPLE xmlns="x-schema:PEOPLE.DTD">

```

The text `xmlns="x-schema:PEOPLE.DTD"` is the pointer to the Schema document against which this XML document will be validated.

XML has already been described in the earlier chapters in this book, so let's now focus on the values and relationships in and between the elements, which is the strength of Schemas. In the example above, there is a group called `<PEOPLE>`, which contains one `<PERSON>`.

You automatically understand that this design or structure is correct by relying on your own experience and how documents like this should be structured. You also know it's OK to have several `<PERSON>` tags within a single `<PEOPLE>` tag, but there should only ever be one `<NAME>` value per position.

A computer (or in a more broad sense, we should say machines) cannot make those judgments, as they have no previous experience to draw on—unless we had already defined which values are acceptable and what the relationships are between the elements. Therefore, when creating a Schema, we define what the acceptable values of each of the tags are and how they relate to each other.

So, what would a Schema for the above PEOPLE XML look like? The following XML Schema would be the minimum Schema needed to enforce that the XML document has the required tags and structure:

```

<?xml version="1.0"?>
<Schema name="people.dtd"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="PEOPLE" content="eltOnly" order="seq">
    <element type="PERSON" minOccurs="1" maxOccurs="*" />
  </ElementType>
  <ElementType name="PERSON" content="eltOnly" order="seq">
    <element type="NAME" />
    <element type="ADDRESS" />
    <element type="TEL" />
    <element type="FAX" />
    <element type="EMAIL" />
  </ElementType>
  <ElementType name="ADDRESS" content="textOnly" />
  <ElementType name="EMAIL" content="textOnly" />
  <ElementType name="FAX" content="textOnly" />
  <ElementType name="NAME" content="textOnly" />

```

```
<ElementType name = "TEL" content = "textOnly"/>
</Schema>
```

Let's pick the Schema above apart a bit so that you can get a sense of some of the exciting features that are available to Schema designers.

```
<Schema name="myschema" xmlns="urn:schema-microsoft-com:xml-data"
  xmlns:dt="urn:schema-microsoft-com:datatype">
<!-- ... -->
</Schema>
```

This is called the *Schema document element*, and it contains the *namespace declarations* that indicate to the parser that the Microsoft built-in data types are being used. Once you have included the Microsoft datatype namespaces, you can prefix your tags with `dt:type`, which allows you to reference the Microsoft data types in the Internet Explorer 5 XML objects.

This Schema document element is called a *top-level declaration*.

```
<ElementType name = "PEOPLE" content = "eltOnly" order = "seq">
<element type = "PERSON" minOccurs = "1" maxOccurs = "*" />
</ElementType>
```

Inside this top-level declaration (called an `ElementType`), we declare the element types that are being used. There is only one, and it has a name of `PERSON`. The `PERSON` element is permitted to have contents of a type called `eltOnly`. Type `eltOnly` means that the element cannot contain any free text, only the specified elements. (These are listed later in this chapter as `Address`, `Email`, `Fax`, `Name`, and `Tel`.)

You can provide more attributes for this element to ensure your data is correctly defined and structured. For example, you can set the number of instances of `PERSON`. This is defined in the `minOccurs` and `maxOccurs` attributes.

```
<element type = "PERSON" minOccurs = "1" maxOccurs = "*" />
```

The order of the elements within `PERSON` (`Address`, `Email`, `Fax`, `Name`, and `Tel`) is set as `seq`, which means the specified elements must appear in the required order.

Already you may be getting the feeling that Schemas can define your data and even outline appropriate user interactions to a certain extent. This is clearly a very useful feature of XML, and properly used, it can lead to massive productivity gains as data and systems begin to describe themselves accurately.

Attributes are very similar to elements, and you would use one over the other because of their different characteristics. The following XML snippet has two elements, `NAME` and `FULL_NAME`:

```
<NAME>
<FULL_NAME>Mark Wilson</FULL_NAME>
</NAME>
```

The element below, `NAME`, has an attribute called `FULL_NAME` and is functionally the same as the preceding snippet:

```
<NAME FULL_NAME="Mark Wilson" />
```

The Microsoft XML-Data Reduced Schemas also provide for a lot of control and definition if we use attributes in our XML. If you used an attribute called `colors`, which is a selection of one of 3 colors, in your XML, an example of the Schema could be:

```
<AttributeType name="colors" dt:type="enumeration" dt:values="red green
blue" required="yes">
```

The attribute above has a name of `colors`, it has a type of enumeration, and the values are `red`, `green`, and `blue`. Lastly, this attribute must have a supplied value.

Although Schemas are very useful and will be used in most instances, they are optional, not required. If your XML application doesn't require validation, you do not need to specify a Schema file. A Schema is an excellent alternative to DTDs, the document that provides similar structure details for SGML and which existed for decades before XML blossomed.

As we discussed before, Schemas have several advantages over DTDs, one of which is the massively improved data typing that is available to you as a developer.

8.3 *Using the data types that are available*

To make use of the data type support included with Internet Explorer 5, your XML Schema must include the `datatype` namespace. The top-level Schema element declaration looks like this:

```
<Schema name="myschema"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <!-- ... -->
</Schema>
```

Within that Schema element, we already know that data can be structured as an `<ElementType>` or `<AttributeType>`. Also, in the Schema element declaration, we used the namespace declarations to indicate that we intend to use the Microsoft data types. We can now use a data type in one of two forms:

```
dt:type attribute
<datatype> element
```

Both forms are demonstrated below as two equivalent and valid samples. This declaration of an `ElementType` called `pages` has a data type of an integer:

```
<ElementType name="pages" dt:type="int"/>
```

This declaration of an `ElementType` is also called “pages” and also has a data type of an integer:

```
<ElementType name="pages">
  <datatype dt:type="int" />
</ElementType
```

The same applies to attributes. The `<datatype>` or `dt:type` attribute can be used directly on the `<AttributeType>` declaration.

Although XML Schema support in Internet Explorer 5 allows data types to be specified within attributes, only the following data types are supported within attributes by the parser and DOM: `string`, `id`, `idref`, `idrefs`, `nmtoken`, `nmtokens`, `entity`, `entities`, `enumeration`, and `notation`. Support for the other data types will be added in a future release.

Table 8.2 provides a complete list of data types supported by the Microsoft DOM Objects. This list can be found at <http://msdn.microsoft.com/xml/reference/schema/datatypes.asp>

8.3.1 Full list of Microsoft data types supported

Table 8.2 Datatypes that are supported

Datatype	Description
<code>bin.base64</code>	MIME-style Base64 encoded binary BLOB
<code>bin.hex</code>	Hexadecimal digits representing octets
<code>boolean</code>	0 or 1, where 0 == "false" and 1 == "true"
<code>char</code>	String, one character long
<code>date</code>	Date in a subset ISO 8601 format, without the time data. For example: "1994-11-05".
<code>dateTime</code>	Date in a subset of ISO 8601 format, with optional time and no optional zone. Fractional seconds can be as precise as nanoseconds. For example, "1988-04-07T18:39:09".
<code>dateTime.tz</code>	Date in a subset ISO 8601 format, with optional time and optional zone. Fractional seconds can be as precise as nanoseconds. For example: "1988-04-07T18:39:09-08:00".
<code>fixed.14.4</code>	Same as "number" but no more than 14 digits to the left of the decimal point, and no more than 4 to the right
<code>float</code>	Real number, with no limit on digits; can potentially have a leading sign, fractional digits, and optionally an exponent. Punctuation as in U.S. English. Values range from 1.7976931348623157E+308 to 2.2250738585072014E-308.
<code>int</code>	Number, with optional sign, no fractions, and no exponent

Table 8.2 Datatypes that are supported (continued)

number	Number, with no limit on digits; can potentially have a leading sign, fractional digits, and optionally an exponent. Punctuation as in U.S. English. (Values have same range as most significant number, R8, 1.7976931348623157E+308 to 2.2250738585072014E-308.)
time	Time in a subset ISO 8601 format, with no date and no time zone. For example: "08:15:27".
time.tz	Time in a subset ISO 8601 format, with no date but optional time zone. For example: "08:15:27-05:00".
i1	Integer represented in one byte. A number, with optional sign, no fractions, no exponent. For example: "1, 127, -128".
i2	Integer represented in one word. A number, with optional sign, no fractions, no exponent. For example: "1, 703, -32768".
i4	Integer represented in four bytes. A number, with optional sign, no fractions, no exponent. For example: "1, 703, -32768, 148343, -1000000000".
r4	Real number, with no limit on digits; can potentially have a leading sign, fractional digits, and optionally an exponent. Punctuation as in U.S. English. Values range from 3.40282347E+38F to 1.17549435E-38F.
r8	Same as "float". Real number, with no limit on digits; can potentially have a leading sign, fractional digits, and optionally an exponent. Punctuation as in U.S. English. Values range from 1.7976931348623157E+308 to 2.2250738585072014E-308.
ui1	Unsigned integer. A number, unsigned, no fractions, no exponent. For example: "1, 255".
ui2	Unsigned integer, two bytes. A number, unsigned, no fractions, no exponent. For example: "1, 255, 65535".
ui4	Unsigned integer, four bytes. A number, unsigned, no fractions, no exponent. For example: "1, 703, 3000000000".
uri	Universal Resource Identifier (URI). For example, "urn:schemas-microsoft-com:Office9".
uuid	Hexadecimal digits representing octets, optional embedded hyphens that are ignored. For example: "333C7BC4-460F-11D0-BC04-0080C7055A83".

8.3.2 *Primitive types*

The W3C XML 1.0 recommendation also defines enumerated types (notations and enumerations) and a set of tokenized types. These types, defined in the W3C XML 1.0 recommendation, are referred to as *primitive types* within this Microsoft XML documentation.

Table 8.3 lists some of the primitive types defined in Section 3.3.1 of the W3C XML 1.0 Recommendation.

Table 8.3 Supported W3C primitive types

Primitive datatype	Description
entity	Represents the XML ENTITY type
entities	Represents the XML ENTITIES type
enumeration	Represents an enumerated type (supported on attributes only)
id	Represents the XML ID type
idref	Represents the XML IDREF type
idrefs	Represents the XML IDREFS type
nmtoken	Represents the XML NMTOKEN type
nmtokens	Represents the XML NMTOKENS type
notation	Represents a NOTATION type
string	Represents a string type

8.3.3 Supported data type conversions

Table 8.4 lists the data type conversions supported by the Microsoft XML processor in Microsoft Internet Explorer 5.

Table 8.4 Datatype conversions supported by Microsoft XML processors

Datatype	Variant type
string	VT_BSTR, VT_BOOL, VT_CY, VT_DATE, VT_VARIANT, VT_DECIMAL, VT_UII, VT_I2, VT_I4, VT_R4, VT_R8, VT_EMPTY, VT_NULL, VT_ARRAY
number	VT_BSTR, VT_BOOL, VT_CY, VT_VARIANT, VT_DECIMAL, VT_UII, VT_I2, VT_I4, VT_R4, VT_R8, VT_EMPTY, VT_NULL
int	VT_BSTR, VT_BOOL, VT_VARIANT, VT_UII, VT_I2, VT_I4, VT_EMPTY, VT_NULL
float	VT_BSTR, VT_CY, VT_VARIANT, VT_DECIMAL, VT_UII, VT_I2, VT_I4, VT_R4, VT_R8, VT_EMPTY, VT_NULL
fixed.14.4 (currency)	VT_BSTR, VT_CY, VT_VARIANT, VT_DECIMAL, VT_UII, VT_I2, VT_I4, VT_R4, VT_EMPTY, VT_NULL
boolean	VT_BSTR, VT_BOOL, VT_VARIANT, VT_DECIMAL, VT_UII, VT_I2, VT_I4, VT_EMPTY, VT_NULL
dateTime	VT_BSTR, VT_DATE, VT_VARIANT
i1	VT_BSTR, VT_BOOL, VT_VARIANT, VT_DECIMAL, VT_EMPTY, VT_NULL
i2	VT_BSTR, VT_BOOL, VT_VARIANT, VT_DECIMAL, VT_I2, VT_EMPTY, VT_NULL

Table 8.4 Datatype conversions supported by Microsoft XML processors (continued)

i4	VT_BSTR, VT_BOOL, VT_VARIANT, VT_DECIMAL, VT_I2, VT_I4, VT_EMPTY, VT_NULL
i8	VT_BSTR, VT_BOOL, VT_CY, VT_VARIANT, VT_DECIMAL, VT_I2, VT_I4
ui1	VT_BSTR, VT_BOOL, VT_VARIANT, VT_DECIMAL, VT_UI1, VT_EMPTY, VT_NULL
ui2	VT_BSTR, VT_BOOL, VT_VARIANT, VT_UI1, VT_I2, VT_EMPTY, VT_NULL
ui4	VT_BSTR, VT_BOOL, VT_VARIANT, VT_UI1, VT_I2, VT_I4, VT_EMPTY, VT_NULL
ui8	VT_BSTR, VT_BOOL, VT_CY, VT_VARIANT, VT_DECIMAL, VT_UI1, VT_I2, VT_I4, VT_EMPTY, VT_NULL
r4	VT_BSTR, VT_CY, VT_VARIANT, VT_DECIMAL, VT_I2, VT_I4, VT_R4, VT_EMPTY, VT_NULL
r8	VT_BSTR, VT_CY, VT_VARIANT, VT_DECIMAL, VT_I2, VT_I4, VT_R4, VT_R8, VT_EMPTY, VT_NULL
uuid	VT_BSTR, VT_VARIANT
uri	VT_BSTR, VT_VARIANT
bin.hex	VT_BSTR, VT_VARIANT, VT_ARRAY
char	VT_BSTR, VT_VARIANT, VT_I2, VT_EMPTY, VT_NULL

8.4 *BizTalk, where it's all happening!*

With BizTalk you will be able to submit your XML-described messages (such as a subscription request or a just-in-time purchase order) in many different ways and from any platform. You can use Microsoft Message Queue Server, the SMTP email protocol, the Internet's HTTP protocol, various COM interfaces, and even FTP, to name just a few ways to get your information into and receive it out of BizTalk.

BizTalk is also aware of existing data storage systems and ERP systems such as SAP and can exchange messages and routing documents with them automatically.

You have to use some predefined XML tags and the associated BizTalk namespace within your XML. Let's take a look at how you do that.

8.4.1 *A BizTalk XML example*

So, what does a BizTalk Framework XML document look like? Below is a simple example of a generic BizTalk Framework data document.

```
<?xml version="1.0"?>
<BizTalk xmlns:="urn:schemas-biztalk.org: BizTalk/biztalk-0.8.xml">
<Body xmlns:= "urn:your-namespace-goes-here">
<Route>
<From Location ID="value" LocationType="value" Process="value"
Path="value" Handle="value"/>
```

```
<To Location ID="value" LocationType="value" Address="value" Path="value"
Handle="value" />
</Route>
<MessageType>
```

-- Your XML document data goes here --

```
</MessageType>
</Body>
</BizTalk>
```

Also, you can see that all the BizTalk tags relate to message exchange and do not include industry-specific tags. This makes the BizTalk Framework industry-neutral and suitable for widespread use.

Let's take a closer look at the tags your BizTalk XML document must use.

The root of the matter

A BizTalk Framework document must begin and end with the BizTalk XML “root” tag:

```
<BizTalk xmlns="urn:schemas-biztalk-org: BizTalk/biztalk-0.8.xml">
</BizTalk>
```

If you take a close look, you will see that a BizTalk namespace is used.

Body and identity tags

The <MessageType> tag shown is a placeholder for the actual document type name tag. It could actually hold a tag for a Purchase Order, Flight, or other entity. The XML document would be marked up with XML tags consistent with the XML-Data proposal.

```
<Body>
<MessageType xmlns="urn:your-namespace-here">
</MessageType>
</Body>
```

Note that this is where your Schema is inserted.

Routing tags

Two types of routing tags are defined in the BizTalk Framework. These are the To and From tags that are required to support point-to-point message exchanges. These To and From tags take the general form described below:

```
<Route>
<From locationID="11111111" locationType="DUNS"
process="" path="" handle="3"/>
<To locationID="22222222" locationType="DUNS"
process="" path="" handle="23CF15"/>
</Route>
```

So, now that you have had a taste of Schemas, and you may have a good grasp of BizTalk and the impact it will have on automated messaging between systems, you may wonder *what's next?*

8.4.2 *Get your schemas here!*

Ah, yes, the million dollar question is: does anyone in your industry already have a Schema for certain areas or concerns in your industry? At the BizTalk website, <http://www.biztalk.org/>, the Microsoft website builders are building a library. It is a catalog with accompanying features that will help you locate Schemas that others have registered and cataloged. Members will be given the opportunity to register their organizations and establish publishing rights.

8.4.3 *Cool tools and websites*

Go to <http://www.biztalk.org>, <http://www.xml.com>, and <http://www.xmltree.com> for more information on your particular industry. You can also take a look at these business-to-business websites that are taking the lead in XML documents and communication, <http://www.marketsite.net> and <http://www.commerceone.com>, which is marketing itself as “Industry’s First Comprehensive XML Document Library.” For more information, sample code, or to find other developers who program with XML or the BizTalk Framework Toolkit, visit <http://www.vbxml.com>

For your own interest, there are also at least five more Schema formats, which are also trying to take over DTDs:

- DCD (Document Content Description), <http://www.w3.org/TR/NOTE-dcd>
- XML-Data Reduced (XDR), <http://www.w3.org/TR/1998NOTE-XML-data/> and <http://msdn.microsoft.com/xml/XMLGuide/schema-overview.asp>
- DDML (also known as XSchema) <http://www.w3.org/TR/NOTE-ddml>
- W3C XML Schema Definition Language (XSDL), <http://www.w3.org/TR/xmlSchema-1/>, <http://www.w3.org/TR/xmlSchema-2/>, and <http://www.w3.org/TR/xmlschema-2/>
- Schema for Object-oriented XML (SOX), <http://www.w3.org/TR/NOTE-SOX/>

Of course, building Schemas would be easier if there were a visual tool that could help you, right? A company called Extensibility, <http://www.extensibility.com/>, has just that kind of tool. XML Authority is their product, and it presents Schemas in a visual way so you can set your constraints visually without getting stuck in the syntax. Even better, it handles all the different Schema proposals essentially the same.

XML Authority is aware of all the different types of Schemas. It does not force you to use any particular one because it is a visual design tool that allows you to “save as” the different types of Schemas.

8.4.4 How different are all the Schemas?

Well, to answer that, let's look at some examples. In this book, we have been using the following XML code snippet:

```
<?xml version="1.0" ?>
<PEOPLE>
<PERSON>
<NAME>Mark Wilson</NAME>
<ADDRESS>911 Somewhere Circle, Canberra, Australia</ADDRESS>
<TEL>(++612) 12345</TEL>
<FAX>(++612) 12345</FAX>
<EMAIL>Mark.Wilson@somewhere.com</EMAIL>
</PERSON>
</PEOPLE>
```

DTD example

For that XML code, this is the correct DTD structure:

```
<!ELEMENT PEOPLE ( PERSON+ ) >
<!ELEMENT PERSON ( NAME, ADDRESS, TEL, FAX, EMAIL ) >
<!ELEMENT ADDRESS ( #PCDATA ) >
<!ELEMENT EMAIL ( #PCDATA ) >
<!ELEMENT FAX ( #PCDATA ) >
<!ELEMENT NAME ( #PCDATA ) >
<!ELEMENT TEL ( #PCDATA ) >
```

Here we investigate the five Schemas that XML Authority generates:

- W3C Schema
- Microsoft XML-data reduced
- Commerce One
- DCD
- DDML

W3C Schema (XSDL) example

For the W3C Schema definition, you will use:

```
<?xml version="1.0"?>
<!DOCTYPE schema PUBLIC "-//W3C//DTD XMLSCHEMA 19990506//EN
"http://www.w3.org/1999/05/06-xmlschema-1/structures.dtd">
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/
TR/xmlschema-1/-->
<schema name="people.dtd"
xmlns="http://www.w3.org/1999/05/06-xmlschema-1/structures.xsd">
<elementType name="PEOPLE">
<sequence>
<elementTypeRef name="PERSON" minOccur="1" maxOccur="*" />
</sequence>
```

You can download your own free trial copy of XML Authority at <http://www.extensibility.com/>

```

</elementType>
<elementType name = "PERSON">
  <sequence>
    <elementTypeRef name = "NAME"/>
    <elementTypeRef name = "ADDRESS"/>
    <elementTypeRef name = "TEL"/>
    <elementTypeRef name = "FAX"/>
    <elementTypeRef name = "EMAIL"/>
  </sequence>
</elementType>
<elementType name = "ADDRESS">
  <datatypeRef name = "string"/>
</elementType>
<elementType name = "EMAIL">
  <datatypeRef name = "string"/>
</elementType>
<elementType name = "FAX">
  <datatypeRef name = "string"/>
</elementType>
<elementType name = "NAME">
  <datatypeRef name = "string"/>
</elementType>
<elementType name = "TEL">
  <datatypeRef name = "string"/>
</elementType>
</schema>

```

Microsoft XML-data reduced (XDR) Schema example

For this Schema definition, you will use:

```

<?xml version = "1.0"?>
<!--Generated by XML Authority. Conforms to XML Data subset for IE 5-->
<Schema name = "people.dtd"
  xmlns = "urn:schemas-microsoft-com:xml-data"
  xmlns:dt = "urn:schemas-microsoft-com:datatypes">
  <ElementType name = "PEOPLE" content = "eltOnly" order = "seq">
    <element type = "PERSON" minOccurs = "1" maxOccurs = "*" />
  </ElementType>
  <ElementType name = "PERSON" content = "eltOnly" order = "seq">
    <element type = "NAME" />
    <element type = "ADDRESS" />
    <element type = "TEL" />
    <element type = "FAX" />
    <element type = "EMAIL" />
  </ElementType>
  <ElementType name = "ADDRESS" content = "textOnly" />
  <ElementType name = "EMAIL" content = "textOnly" />
  <ElementType name = "FAX" content = "textOnly" />
  <ElementType name = "NAME" content = "textOnly" />
  <ElementType name = "TEL" content = "textOnly" />

```

```
</Schema>
```

Commerce One SOX Schema example

For this Schema definition, you will use:

```
<?xml version = "1.0"?>
<!DOCTYPE schema SYSTEM "schema.dtd">
<!--Generated by XML Authority. Conforms to w3c NOTE-SOX-19980930-->
<schema name = "people.dtd">
<h1>people.dtd</h1>
<elementtype name = "PEOPLE">
  <model>
    <sequence>
      <element name = "PERSON" occurs = "+"/>
    </sequence>
  </model>
</elementtype>
<elementtype name = "PERSON">
  <model>
    <sequence>
      <element name = "NAME"/>
      <element name = "ADDRESS"/>
      <element name = "TEL"/>
      <element name = "FAX"/>
      <element name = "EMAIL"/>
    </sequence>
  </model>
</elementtype>
<elementtype name = "ADDRESS">
  <string/>
</elementtype>
<elementtype name = "EMAIL">
  <string/>
</elementtype>
<elementtype name = "FAX">
  <string/>
</elementtype>
<elementtype name = "NAME">
  <string/>
</elementtype>
<elementtype name = "TEL">
  <string/>
</elementtype>
</schema>
```

DCD Schema example

For this Schema definition, you will use:

```
<?xml version = "1.0"?>
<!DOCTYPE DCD SYSTEM "tbd">
```

```

<!--Generated by XML Authority. Conforms to DCD 1.0-->
<DCD xmlns:RDF = "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<ElementDef type = "PEOPLE" Content = "Closed" Model = "Elements">
  <Group RDF:Order = "Seq">
    <Group Occurs = "OneOrMore">
      <Element>PERSON</Element>
    </Group>
  </Group>
</ElementDef>

<ElementDef type = "PERSON" Content = "Closed" Model = "Elements">
  <Group RDF:Order = "Seq">
    <Element>NAME</Element>
    <Element>ADDRESS</Element>
    <Element>TEL</Element>
    <Element>FAX</Element>
    <Element>EMAIL</Element>
  </Group>
</ElementDef>

<ElementDef type = "ADDRESS" Content = "Closed" Model = "Data"/>
<ElementDef type = "EMAIL" Content = "Closed" Model = "Data"/>
<ElementDef type = "FAX" Content = "Closed" Model = "Data"/>
<ElementDef type = "NAME" Content = "Closed" Model = "Data"/>
<ElementDef type = "TEL" Content = "Closed" Model = "Data"/>
</DCD>

```

DDML Schema example

For this Schema definition, you will use:

```

<?xml version ="1.0"?>
<!DOCTYPE DocumentDef [
]>
<!--Generated by XML Authority. DDML version 1.0-->
<DocumentDef name = "people.dtd"
  xmlns = "http://www.purl.org/NET/ddml/v1"
  xmlns:DDML = "http://www.purl.org/NET/ddml/v1" Version = "1.0">
<ElementDecl Name = "PEOPLE">
  <Model>
    <Ref Element = "PERSON" Frequency = "OneOrMore"/>
  </Model>
</ElementDecl>
<ElementDecl Name = "PERSON">
  <Model>
    <Seq>
      <Ref Element = "NAME"/>
      <Ref Element = "ADDRESS"/>
      <Ref Element = "TEL"/>
      <Ref Element = "FAX"/>
      <Ref Element = "EMAIL"/>
    </Seq>
  </Model>
</ElementDecl>

```

```
    </Seq>
  </Model>
</ElementDecl>
<ElementDecl Name = "ADDRESS">
  <Model>
    <PCData/>
  </Model>
</ElementDecl>
<ElementDecl Name = "EMAIL">
  <Model>
    <PCData/>
  </Model>
</ElementDecl>
<ElementDecl Name = "FAX">
  <Model>
    <PCData/>
  </Model>
</ElementDecl>
<ElementDecl Name = "NAME">
  <Model>
    <PCData/>
  </Model>
</ElementDecl>
<ElementDecl Name = "TEL">
  <Model>
    <PCData/>
  </Model>
</ElementDecl>
</DocumentDef>
```

8.5 Summary

With potentially as many as 56 million places to visit on the Internet, you may be thinking the words *information overload*. Frequently we wonder how we will be able to find information we need in all that unstructured information out there. But with the growing use of XML, DTDs, RDF, RDF Schemas, and the various versions of XML Schemas, we can look forward to better software and a more accurate understanding of that information.

But that's not all. We can also look forward to businesses sharing their Schemas and (hopefully) using the same ones, or at least mapping their Schemas to each other. When this happens, transparent data transfer, automatic billing and reconciliation, and loads of other things will begin to make our lives easier.

The XML/Schema/BizTalk combination is undoubtedly a knockout punch and will help usher in widespread, inexpensive, and worldwide use of eCommerce. Its potential to provide a platform for business-to-business exchange is enormous.

However, as XML travels across this sea of information, it may become more like a modern-day Gulliver than a Christopher Columbus. While Columbus discovered the New World, Gulliver became tied down by thousands of bit-part players. The same thing could happen to XML if poorly defined Schemas become prevalent and are used in document repositories such as BizTalk.

Since the jury is still out on where all of this is going, and since the Schema specification is not finalized, we have not focused on using Schemas in this book. We have been using DTDs, as they are a mature, easy-to-use, and stable standard. Going from DTDs to Schemas is not a complex step, and you will be able to make your own decision on which route to take.