Schemas

Objectives:

- To understand what a schema is;
- To understand the differences between DTDs and schema;
- To be able to create Microsoft XML schema;
- To use schema to describe elements and attributes;
- To use schema data types.

Document Type Definitions (DTDs) describe an XML document's structure. DTDs are inherited from SGML. Many developers in the XML community feel DTDs are not flexible enough to meet today's programming needs:

- DTDs cannot be manipulated (e.g., searched, transformed into different representation such as HTML, etc.) in the same manner as XML documents can because DTDs are not XML documents.
- Schemas are the alternative to DTDs, for validating XML documents. Like DTDs, schemas must be used with validating parsers. Schemas are expected to replace DTDs as the primary means of describing document structure.
- Two major schema models exist: W3C XML schema and Microsoft XML schema. Because W3C XML schema technology is still in the early stages of development, the focus is primarily on the well-developed Microsoft XML schema.

Observations:

- Schema documents use XML syntax and are therefore XML documents;
- Schemas are XML documents that conform to DTDs, which define the structure of a schema. These DTDs are bundled with the parser and are used to validate the schema that authors create.
Many organizations and individuals are creating DTDs and schemas for a broad range of categories (e.g., financial transactions, medical prescriptions, etc.). These collections - called repositories - are often available free for download from the Web.

Schema vs. DTDs

A few major differences between XML schema and DTDs. A DTD describes an XML document's structure - and not its element content. For example,

```
<quantity>5</quantity>
```

contains character data. Element quantity can be validated to confirm that it does indeed contain content (e.g., PCDATA), but its content cannot be validated to confirm that it is numeric; DTDs do not provide such a capability. So, markup such as

```
<quantity>hello</quantity>
```

is also considered valid. The application using the XML document containing this markup would need to test if quantity is numeric and take appropriate action if it is not. With XML schema, element quantity's data can indeed be described as numeric, so an XML schema that specifies element quantity's data must be numeric,

```
5 conforms and hello fails.
```

Observation:

- Because schema are XML documents that conform to DTDs, they must be valid.

- Unlike DTDs, schema do not use the Extended Backus-Naur Form (EBNF) grammar. Instead, schema use XML syntax.
- Because schema are XML documents, they can be manipulated (e.g., elements added, elements removed, etc.) like any other XML document.
- Document Object Model is to manipulate XML documents programmatically.
Microsoft XML Schema: Describing Elements

Elements are the primary building blocks used to create XML documents. In Microsoft XML Schema, element **ElementType** defines an element.

**ElementType** contains attributes that describe the element's content, data type, name, etc.

Portability:

- *To use Microsoft XML Schema, Microsoft's XML parser (msxml) is required; this parser is part of Internet Explorer 5.*

Element **Schema** can contain only elements:

- **ElementType** - for defining elements,
- **AttributeType** - for defining attributes,
- **description** - for describing the Schema element.

Example 1 presents a complete schema. This schema describes the structure for an XML document that marks up messages passed between users. The schema is named under *intro-schema.xml*:

```
<Schema xmlns = "urn:schemas-microsoft-com:xml-data">
```

declares the Microsoft XML Schema root element. Element **Schema** is the root element for every Microsoft XML Schema document. The **xmlns** attribute specifies the default namespace for the Schema element and the elements it contains. The attribute value *urn:schemas-microsoft-com:xml-data* specifies the URI for this namespace. Microsoft Schema documents always use this URI because it is recognized by msxml. Microsoft's XML parser recognizes element **Schema** and this particular namespace URI and validates the schema.
Example 1: Microsoft XML Schema document

define element message, which can contain only text, because attribute content is textonly. Attribute model has the value closed - indicating that only elements declared in this schema are permitted in a conforming XML document. Any elements not defined in this schema would invalidate the document. Element description contains text that describe this schema. In this particular case it is indicated in the description element that the message element is intended to contain Text messages.
Observation:

*Element description* provides a means for the schema author to provide information about a schema to a parser or application using the schema.

```xml
<ElementType name = "greeting" model = "closed"
    content = "mixed" order = "many">
    <element type = "message"/>
</ElementType>
```

define element *greeting*. Because attribute *content* has the value *mixed*, this element can contain both elements and character data. The *order* attribute specifies the number and order of child elements a *greeting* element may contain. The value *many* indicates that any number of *message* elements and text can be contained in the *greeting* element in any order. The *element* element indicates *message* elements may be included in a *greeting* element.

```xml
<ElementType name = "myMessage" model = "closed"
    content = "eltOnly" order = "seq">
    <element type = "greeting" minOccurs = "0"
        maxOccurs = "1"/>
    <element type = "message" minOccurs = "1"
        maxOccurs = "*"/>
</ElementType>
```

define element *myMessage*. The *content* attribute's value *eltOnly* specifies that the *myMessage* element can only contain elements. Attribute *order* has the value *seq*, indicating that *myMessage* child elements must occur in the sequence defined in the schema.

```xml
<element type = "greeting" minOccurs = "0"
    maxOccurs = "1"/>
<element type = "message" minOccurs = "1"
    maxOccurs = "*"/>
```

indicate that element *myMessage* contains child elements *greeting* and *message*. These elements are *myMessage* child elements, because the *element* elements that reference them are nested inside element *myMessage*. Because the element order in element *myMessage* is set as sequential, the *greeting* element (if used) must precede all *message* elements. Attributes *minOccurs* and *maxOccurs* specify the minimum and maximum number of times the element may appear in the *myMessage* element, respectively. The value 1 for the *minOccurs* attribute indicates that element *myMessage* must contain at least one *message* element. The value *"* for the *maxOccurs* attribute indicates that there is no limit on the maximum number of *message* elements that may appear in *myMessage*. 
<?xml version ="1.0"?>
<myMessage xmlns = "x-schema:intro-schema.xml">
   <greeting>Welcome to XML Schema!
       <message>This is the first message.</message>
   </greeting>
   <message>This is the second message.</message>
</myMessage>

Example 2 shows an XML document that conforms to the schema shown in example 1. Microsoft's XML Validator is used to check the document's conformity. It is available as a free download at msdn.microsoft.com/downloads/samples/internet/xml/xml_validator/sample.asp

<myMessage xmlns = "x-schema:intro-schema.xml">
references the schema through the namespace declaration. A document using a Microsoft XML Schema uses attribute xmlns to reference its schema through a URI which begins with x-schema followed by a colon (:) and the name of the schema document.

<greeting>Welcome to XML Schema!
   <message>This is the first message.</message>
</greeting>

use element greeting to mark up text and a message element. Recall that in example 1, element greeting may contain mixed content.

<message>This is the second message.</message>

marks up text in a message element, the element message can contain only text. In the discussion above, it was mentioned that a closed model allows an XML document to contain only those elements defined in its schema. For example, the markup

<greeting>Welcome to XML Schema!
   <message>This is the first message.</message>
   <newElement>A new element.</newElement>
</greeting>

uses element newElement, which is not defined in the schema. With a closed model, the document containing newElement is invalid. However, with an open model, the document is valid.
Observation:

*The open model makes Microsoft XML Schema documents extensible by allowing authors to add elements to documents without invalidating a document.*

Example 3 shows a well-formed document that fails to conform to the schema of example 1, because element `message` cannot contain child elements.

```xml
<?xml version = "1.0"?>
<myMessage xmlns = "x-schema:intro-schema.xml">
    <greeting>Welcome to XML Schema!</greeting>
    <message>This is message containing another message.
        <message>This is the inner message.</message>
    </message>
</myMessage>
```

Example 3 Well-formed, but invalid XML document.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Describes the element's content. The valid values for this attribute are <strong>empty</strong> (an empty element), <strong>eltOnly</strong> (may contain only elements), <strong>textOnly</strong> (may contain only text) and <strong>mixed</strong> (may contain both elements and text). The default value for this attribute is <strong>mixed</strong>.</td>
</tr>
<tr>
<td><code>dt:type</code></td>
<td>Defines the element's data type. Data types exist for real numbers, integers, booleans, etc. Namespace prefix <code>dt</code> qualities data types.</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>The element's name, a required attribute.</td>
</tr>
</tbody>
</table>
Model  Specifies whether elements not defined in the schema are permitted in the element. Valid values are open (the default, which permits the inclusion of elements defined outside the schema) and closed (only elements defined inside the schema are permitted).

Order  Specifies the order in which child elements must occur. The valid values for this attribute are one (exactly one child element is permitted), seq (child elements must appear in the order in which they are defined) and many (child elements can appear in any order, any number of times). The default value is many if attribute content is mixed and is seq if attribute content has the value eltOnly.

Table 4. ElementType element attributes.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Provides a description of the ElementType.</td>
</tr>
<tr>
<td>datatype</td>
<td>Specifies the data type for the ElementType element.</td>
</tr>
<tr>
<td>element</td>
<td>Specifies a child element by name.</td>
</tr>
<tr>
<td>group</td>
<td>Groups related element elements and defines their order and frequency.</td>
</tr>
<tr>
<td>AttributeType</td>
<td>Defines an attribute.</td>
</tr>
<tr>
<td>attribute</td>
<td>Specifies an AttributeType for an element.</td>
</tr>
</tbody>
</table>

Table 5. Element ElementType's child elements.
If the `content` attribute for an `ElementType` element has the value `eltOnly` or `mixed content`, the `ElementType` may only contain the elements listed in table 5.

**Good Programming Practice:**
- *Although the ElementType attributes have default values, explicitly writing the attribute and its value improves the schema's readability.*
- *Although 1 is the default value for attributes minOccurs and maxOccurs, explicitly writing the attribute and its value improves the schema's readability.*

The element `element` does not define an element, but rather refers to an element defined by an `ElementType`. This allows the schema author to define an element once and refer to it from many places inside the schema document. The attributes of the `element` element are listed in table 6.

As mentioned in table 5, element `group` creates groups of `element` elements. Groups define the order and frequency in which elements appear using the attributes listed in table 7.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>A required attribute, specifies a child element's name (defined in ElementType)</td>
</tr>
<tr>
<td>MinOccurs</td>
<td>Specifies the minimum number of occurrences an element can have. The valid values are 0 (the element is optional) and 1 (the element must occur one or more times). The default value is 1.</td>
</tr>
<tr>
<td>MaxOccurs</td>
<td>Specifies the maximum number of occurrences. The valid values are 1 (the element occurs at most once) and * (the element can occur any number of times). The default value is 1 unless the ElementType's content attribute is mixed.</td>
</tr>
</tbody>
</table>
### Table 6. Element `element` attributes.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>Specifies the order in which the elements occur. The valid values are one (contains exactly one child element from the <code>group</code>), seg (all child elements must appear in the sequential order in which they are listed) and many (the child elements can appear in any order, any number of times).</td>
</tr>
</tbody>
</table>

| minOccurs      | Specifies the minimum number of occurrences an element can have. The valid values are 0 (the element is optional) and 1 (the element must occur at least once), default value is 1. |

| maxOccurs      | Specifies the maximum number of occurrences an element can have. The valid values are 1 (the element occurs at most once) and * (the element can occur any number of times). The default value is 1 unless the `ElementType's content` attribute is mixed. |

### Table 7. Element `group's` attributes.

Microsoft XML Schema: Describing Attributes

- XML elements can contain attributes that describe elements. In Microsoft XML Schema, element `AttributeType` defines attributes. Table 8 lists `AttributeType` element attributes.
- Like element `ElementType` element, element `AttributeType` may contain description elements and datatype elements.
- To indicate that an element has an `AttributeType`, element `attribute` is used. The attributes of the `attribute` element are shown in table 9.
- Figure 7.10 is a schema for a contact list document that contains a person's name, address and phone number(s).
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Specifies the attribute's default value.</td>
</tr>
<tr>
<td>dt:type</td>
<td>Defines the element's data type. Data types exist for real numbers, integers, <em>booleans</em>, <em>enumerations</em> (i.e., a series of values from which one can be selected), etc. Namespace prefix <code>dt</code> qualifies data types.</td>
</tr>
<tr>
<td>dt:values</td>
<td>Contains an enumeration data type's values.</td>
</tr>
<tr>
<td>name</td>
<td>The attribute name. This is a required attribute.</td>
</tr>
<tr>
<td>required</td>
<td>Indicates whether the attribute is required. The valid values for this attribute are <em>yes</em> and <em>no</em>. The default value is <em>no</em>.</td>
</tr>
</tbody>
</table>

**Table 8.** Element **AttributeType**'s attributes.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Specifies the attribute's default value. This value overrides the value defined in the <strong>AttributeType</strong> element.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the name of the <strong>AttributeType</strong> for the attribute. This is a required attribute.</td>
</tr>
<tr>
<td>required</td>
<td>Indicates whether the attribute is required. Valid values for this attribute are <em>yes</em> and <em>no</em>. The default value is <em>no</em>.</td>
</tr>
</tbody>
</table>

**Table 9.** Element **attribute**'s attributes.
<?xml version="1.0"?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data">
    <ElementType name="contact" content="eltOnly"
        order="seq" model="closed">
        <AttributeType name="owner" required="yes"/>
        <attribute type="owner"/>
        <element type="name"/>
        <element type="address1"/>
        <element type="address2" minOccurs="0" maxOccurs="1"/>
        <element type="city"/>
        <element type="state"/>
        <element type="zip"/>
        <element type="phone" minOccurs="0" maxOccurs="*"/>
    </ElementType>
    <ElementType name="name" content="textOnly"
        model="closed"/>
    <ElementType name="address1" content="textOnly"
        model="closed"/>
    <ElementType name="address2" content="textOnly"
        model="closed"/>
    <ElementType name="city" content="textOnly"
        model="closed"/>
    <ElementType name="State" content="textOnly"
        model="closed"/>
    <ElementType name="zip" content="textOnly"
        model="closed"/>
    <ElementType name="phone" content="textOnly"
        model="closed"/>
</Schema>
Example 10: Demonstrating AttributeType and attribute.

<Schema xmlns = "urn:schemas-microsoft-com:xmi-data"> specifies the default namespace for the URI.

<AttributeType name = "owner" required = "yes"/>
<attribute type = "owner"/> define the contact element attribute owner. The AttributeType element defines the properties of the attribute (e.g., its name). An attribute element creates an attribute with a specific AttributeType for an element. The name of the AttributeType is referenced in the type attribute of the attribute element. In this particular case, line 12 indicates that element contact has an owner attribute.

<ElementType name = "phone" content = "textOnly" model = "closed">
<AttributeType name = "location" default = "home"/>
<attribute type = "location"/>
</ElementType> define element phone, which can contain textOnly and has one attribute named location. If location is omitted, "home" is the default. An XML document that conforms to the contact list schema is shown in example 11.

<?xml version = "1.0"?>
<contact owner = "Bob Smith" xmlns = "x-schema:contact-schema.xml">
<name>Jane Doe</name>
<address>123 Main St.</address>
<city>Sometown</city>
<state>somestate</state>
<zip>12345</zip>
<phone>617-555-1234</phone>
<Phone location = "work">978-555-4321</Phone>
</contact>

Example 11: Contact list that conforms to
Microsoft XML Schema: Data Types

One important schema feature is the ability to indicate the type of content an element or attribute contains—something not possible with DTDs, which treat element and attribute content as text.

To use data types, namespace prefix `dt` is defined (by the document author) and assigned the URI `urn:schemas-microsoft-com:datatypes`. A wide variety of data types exists, some of which are listed in example 12. For a complete list of data types visit:  

`msdn.microsoft.com/xml/reference/schem/datatypes.asp`

**Programming Practice:**

*By convention, Microsoft XML Schema authors use namespace prefix `dt`: when referring to the URI `urn:schemas-microsoft-com:datatypes`.*

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>0 (false) or 1 (true).</td>
</tr>
<tr>
<td>char</td>
<td>A single character (e.g., &quot;D&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>A series of characters (e.g., &quot;Deitel&quot;)</td>
</tr>
<tr>
<td>float</td>
<td>A real number (e.g. 123.4567890)</td>
</tr>
<tr>
<td>int</td>
<td>A whole number (e.g. 5).</td>
</tr>
<tr>
<td>date</td>
<td>A date formatted as yyyy - mm - dd</td>
</tr>
<tr>
<td>time</td>
<td>A time formatted as hh - mm - ss</td>
</tr>
<tr>
<td>id</td>
<td>Text that uniquely identifies an element or attribute.</td>
</tr>
<tr>
<td>idref</td>
<td>A reference to an id</td>
</tr>
<tr>
<td>enumeration</td>
<td>A series of values from which only one may be chosen.</td>
</tr>
</tbody>
</table>

**Table 12:** Some Microsoft XML Schema data types
Example 13 is a schema for an XML document containing book shipping information.

```xml
<?xml version = "1.0"?>
<Schema xmlns = "urn:schemas-microsoft-com:xml-data"
xmlns:dt = "urn:schemas-microsoft-com:datatypes">
<ElementType name = "bookstore" content = "eltOnly"
    order = "many" model = "closed">
    <element type = "shipping"/>
    <element type = "book"/>
</ElementType>
<ElementType name = "shipping", content = "eltOnly"
    order = "seg" model = "closed">
    <AttributeType name = "shipID" dt:type = "id"
        required = "yes"/>
    <attribute type = "shipID"/>
    <element type = "duration"/>
</ElementType>
<ElementType name = "duration" content = "textOnly"
    model = "closed" dt:type = "date"/>
<ElementType name = "book" content = "textOnly"
    model = "closed" dt:type = "string">
    <AttributeType name = "ShippedBy" dt:type = "idref"/>
    <attribute type = "shippedBy"/>
</ElementType>
</Schema>
```

**Example 13:** Using Microsoft XML Schema data types

`<AttributeType name = "shipID" dt:type = "id" required = "yes"/>` assigns attribute `dt: type` the value `id`. This defines attribute `shipID` as the unique identifier for element `shipping`.

`<ElementType name = "book" content = "textonly" model = "closed" dt:type = "string">` define element `book`, which can contain only text. This element's `content` has data type `string`. 
<AttributeType name = "ShippedBy" dt:type = "idref"/> specifies attribute shippedby's data type as idref - which indicates that attribute shippedby must be assigned an attribute declared with type id.

Example 14 shows an XML document that conforms to example 13's schema.

```xml
<?xml version = "1.0"?>
<bookstore xmlns = "x-schema:id-schema.xml">
  <shipping shipID = "s1">
    <duration>2000-08-01</duration>
  </shipping>
  <shipping shipID = "s2">
    <duration>2000-08-20</duration>
  </shipping>
  <book shippedBy = "s1">
    Java How to Program 3rd edition.
  </book>
  <book shippedBy = "s2">
    C How to Program 3rd edition.
  </book>
  <book shippedBy = "s2">
    C++ How to Program 3rd edition.
  </book>
</bookstore>
```

**Example 14: XML document conforming to id-schema.xml**

The shipping elements contain unique identifiers s1 and s2. Each of these elements contains a duration element that marks up a date. The book elements use the shippedBy attribute to reference the unique identifiers s1 and s2. Recall that the schema requires attribute shippedBy to have an idref data type.

- **Microsoft XML Schema:**
  sdn.microsoft.com/xml/reference/schema/start.asp

Internet and World Wide Web Resources:

msdn.microsoft.com/xml/xmlguide/schema-overview.asp

*The Microsoft Schema Developer's Guide* provides an extensive coverage of schemas from a basic introduction to advanced definitions and uses.

msdn.microsoft.com/xml/reference/schenta/start.asp

*The Microsoft XML Schema Reference* contains an introduction to schema.

msdn.microsoft.com/downloads/samples/internet/xml/xml validator/sample.asp

*The Microsoft XML Schema Validator* can be used with both schemas and DTDs.

msdn.microsoft.com/xml/reference/schema/datatypes.asp

Provides a list of *Microsoft XML Schema* data types.

www.w3.org/XML/Schema

The W3C XML Schema Primer - provides a succinct introduction to W3C XML Schemas.

www.DTD.com

Provides a schema/DTD repository for a wide range of technologies. Provides a Web-based program named *DTDFactorv* from which DTDs can be created, edited and saved to disk. DTDFactory allows DTD authors to submit DTDs they create to the repository. In the future, a Web-based program named *SchemaFactory* will be implemented for creating and editing W3C Schema.

www.xmlspy.com/download.html

Download for XML Spy 3.0 development environment. A 30-day trial version is available at this site.

www.extensibility.com

Home page for the *XML Authority* W3C Schema and DTD editor.

www.xmlschema.com

Provides various resources for creating and validating W3C schema.

**SUMMARY**

• A DTD cannot be manipulated (e.g., searched, transformed into a different representation such as
HTML, etc.) in the same manner as an XML document - because it is not an XML document.

- XML schemas are an alternative for validating XML documents. DTDs and schemas need validating parsers. Schemas are an emerging technology that is expected to eventually replace DTDs as the primary means of describing XML document structure.

- Two major schema models exist: W3C XML Schema and Microsoft XML Schema. W3C XML Schema is still in the early stages of development; Microsoft XML Schema is well defined.

- An XML document that conforms to a schema document is schema valid; a document that does not conform is invalid.

- Schema do not use the Extended Backus-Naur Form (EBNF) grammar, but use XML syntax and can be manipulated (e.g., elements can be added, removed, etc.) as with any other XML document.

- Elements are the primary building blocks used to create XML documents. In Microsoft XML Schema, element ElementType defines elements. Because it is an element, ElementType may contain attributes that describe the element's content, data type, name, etc.
• Element element does not define an element, but rather refers to an element defined by ElementType. This allows the schema author to define an element once and refer to it from potentially many places inside the schema document.

• Element Schema is the root element for every Microsoft XML Schema document. Keyword xmlns specifies the namespace (i.e., urn: schemas-microsoft-com:xml-data) used by Schema and its child elements.

• Microsoft's XML parser automatically recognizes element Schema and its namespace URI, and validates the schema. Element Schema can contain elements ElementType - for defining elements, AttributeType - for defining attributes and description - for describing the Schema element.

• Attribute content specifies the content allowed in an element or attribute.

• Attribute model specifies whether elements can contain elements not defined in the schema. Any elements not defined in the schema invalidate the document. Assigning model a value of open indicates that definitions not defined in the schema are permitted. Assigning model a value of closed specifies that elements defined only in the schema are permitted.
- Element description contains any text the document author chooses to describe the schema document.

- When attribute content is mixed, an element can contain both elements and character data.

- When attribute order is assigned many, any number of child elements and text can be combined - in any order.

- When attribute order is assigned seq, child elements must occur in sequential order.

- A document using a Microsoft XML Schema references the schema through a URI that begins with x-schema, followed by a colon (:) and the name of the schema document.

- XML elements can contain attributes that provide additional information for describing elements. In Microsoft XML Schema, element AttributeType defines attributes.

- Element AttributeType can contain description elements and datatype elements.

- To indicate that an element has an AttributeType, element attribute is used.

- One important schema feature is the ability to indicate the type of content an element or attribute has -
something not possible with DTDs, which treat content as text.

- To use data types in Microsoft XML Schema, namespace prefix dt is defined and assigned the URI urn: schemas-microsoft-com: datatypes. A wide variety of data types exists.

- The W3C is developing an XML Schema specification, which is at the time a W3C Candidate Recommendation.

- Although virtually any extension may be used, Microsoft XML Schema documents commonly use the .xml extension and W3C XML Schema documents commonly use the .xsd extension.


- In W3C XML Schema documents, element element defines elements. Attributes name and type specify the element's name and data type, respectively. Attribute ref references an existing element definition.

- When an element has a type such as string (in W3C XML Schema), it is prohibited from containing attributes and child elements. Any element that contains attributes or child elements, must define a type - called a complex type - that defines each attribute and child element.