XML Schema Essentials

R. Allen Wyke
Andrew Watt
Contents

Introduction ix
Acknowledgments xi
About the Authors xiii

Part One Getting Started 1

Chapter 1 Elementary XML Schema 3
How Does an XML Schema Processor Work? 6
What Is XML Schema for? 7
XSD Schema Schema Components 7
Other Schema Languages for XML 8
The DTD Descended from SGML 8
XSD Schema Tools 9
XML Schema Document 14
### Contents

Root of an XML Schema Document 16  
Declaring the Location of Your XML Schema Document 16  
Declaring Elements and Defining Types 17  
  Defining Simple Types 17  
  Defining Complex Types 25  
    Anonymous Complex Types 25  
    Named Complex Types 26  
  Using Anonymous or Named Complex Types 29  
  Declarations 34  
Annotations in Schema 42  
  Standard XML Comments 42  
  The <annotation> Element 43  
Empty Element Declaration 45  
  The anyType Type 50  
Occurrence Constraints 51  
  Cardinality in DTDs 51  
  minOccurs and maxOccurs 51  
  Defining Your Own Simple Type 56  
Model Groups in Schema 57  
  Sequence Group 57  
  Choice Group 57  
  All Group 58  
  Attribute Groups 61  
  More about the XML 1.0 DTD Content Model 62  
Validation in XSD Schema 63  
  Validation versus Assessment 64  
  XML Information Set 67  
  Post-Schema Validation Infoset 69  
Summary 69  

**Chapter 2**  
**XSD Elements** 71  
XML Elements 72  
  Defining within a DTD 72  
  Limitations 76
Chapter 7  Grouping Elements and Attributes 239
Reusing Definitions with Groups 240
   Nesting Sequence Groups 245
   Nesting Choice Groups 246
Substitution Groups 250
Attribute Groups 257
Summary 258

Chapter 8  Deriving Types 259
Deriving Types by Extension 260
Deriving Types by Restriction 271
The enumeration Element 279
The pattern Element 282
The xsi:type Attribute 284
Summary 285

Part Three  Next Steps 287

Chapter 9  Uniqueness and Keys in XSD Schema 289
Identity-Constraint Definitions 289
The <xsd:unique> Element 290
The <xsd:key> and <xsd:keyref> Elements 298
Summary 303

Chapter 10  Bringing the Parts Together 305
Modularizing Schemas 305
   How to Use Schema Modules 306
Creating the Example 325
Planning the Example 325
   Defining the Information Needs 325
   Documenting the Schema 326
   Basic Schema Templates 327
   Modularizing the Schemas 328
Back in February 1998, XML 1.0 was released among the most hype and media coverage that the Internet community had seen since the first version of Java. XML was supposed to solve many of the problems that existed in heterogeneous environments, where applications were forced to communicate and exchange data in proprietary formats. The explosion of the Web had introduced the common HTML format for marking up and exchanging documents, but the structure and potential of HTML to be more than that simply did not exist.

XML, whose foundation was based on SGML, provided a means for people, companies, or entire industries to define languages that could be used to mark up data in a method that others could support and understand. Simply conforming to the well-formed and valid (which is technically optional) requirements of XML was a huge step, and if you coupled that with inherit structure of document type definitions (DTD), users were able to provide a wealth of knowledge to partners with whom they exchanged data. XML offered some datatyping, however, and did not really support a more flexible means of defining schemas.

To help accommodate these deficiencies, other standards such as Datatypes for DTDs (DT4DTD), Schema for Object-Oriented XML (SOX), XML Data, and
Document Definition Markup Language (DDML) were developed and combined with XML data for exchanges. But while these provided many of the features that users needed, integrating multiple standards were cumbersome and less desired than a single, standard approach. Enter XML Schema (XSD).

XSD, which was inspired by all the previously mentioned standards, does not necessarily replace XML—but in many senses of the word, it can be thought of as XML on steroids. It can be the perfect solution for large solutions that include many various types of data integration. When you have applications or entire systems that need to communicate yet have very diverse methods of storing data, XSD can act as the bridge between these systems. These complex solutions need more, and XSD offers that.

What to Expect

In XML Schema Essentials, our job as authors is to expose you to the various publications that are part of the XSD Recommendations. For those of you who have attempted to read and study the recommendations, you know that it can be complex and hard to follow. But just knowing and understanding the standard is only half the battle. We will also expose you to using it to solve real-world problems as well as have discussions about best practices and how you can get the most out of your implementation.

Our goal is simple: for you to finish this book and not only understand XSD but also understand what you can do with it.

Book Organization

In our attempt to teach you XSD, we have taken the approach of stepping through the recommendations from a functional standpoint rather than from top to bottom. The book itself is divided into four parts. The first part, “Getting Started,” introduces you to XSD. You will learn the basic concepts, how to define elements, and how to add attributes to those elements.

Part Two, “Going beyond DTDs,” will focus on functionality that is open and beyond that found in XML DTDs. You will learn about datatypes and how to derive your own datatypes. There are also a couple of chapters that focus on data facets, which are ways you can constrain things such as datatypes. There is also a chapter on grouping elements and attributes. One of the things you will quickly learn about XSD is that you can define more than one root element.

The third part of the book, “Next Steps,” is just that: next steps. In the final two chapters of the book, which are contained in this section, you will learn about some advanced topics that revolve around the use of XSD schemas and
essentially expose yourself to a deeper level of topics than covered in previous chapters. You will also work through an example that ties together everything you have learned up until this point to result in a full understanding of XSD.

Finally, in Part Four, which contains Appendixes A and B, we have included a reference for both the datatypes (primitive and derived) and the facets available in the XSD Recommendations. We hope that you will use the material contained here even after you have finished reading the book, because it can serve as a valuable reference.

A Final Thought

This brief introduction should basically prepare you for what to expect from the pages that follow. We did not want to waste your time here rambling on about random thoughts of how XSD will solve the world’s problems. Simply put, we want you to come to your own conclusions. So, we have saved our discussion of why and how XSD could possibly do so, at least in the computing world, for the chapters and pages within the book itself.

R. Allen Wyke
Andrew Watt

Acknowledgments

R. Allen Wyke

On the publishing side, I would like to thank Bob Kern of TIPS Publishing and my co-author, Andrew, for their professionalism, hard work, and overall support in the proposing and writing of this book. I would also like to thank all the people at Wiley who worked on the book and helped make sure it was the best it could be.

Andrew Watt

I would like to thank my co-author, Allen, for his contribution to the development and writing of this book. Thanks, too, to Scott Amerman, Penny Linskey, and the team at Wiley for doing all that was necessary to bring this book to fruition.
I would like to dedicate this book to the citizens of New York City, the United States of America, and the world for their perseverance and strength following the tragic events that occurred September 11, 2001.

R. Allen Wyke

I would like to dedicate this book to the memory of my late father, George Alec Watt, a very special human being.

Andrew Watt
About the Authors

R. Allen Wyke

R. Allen Wyke of Durham, North Carolina is the Vice-President of Technology at Blue292, a pioneering company on the forefront of environment, health, safety, and emergency management software and services. At Blue292, he works with management and engineering to help ensure and create products that have the proper vision and direction while fulfilling customers’ expectations. He is constantly working with Java, XML, JavaScript, and other related Internet technologies—all of which are part of the framework used for the Blue292 systems.

Allen, who wrote his first computer program at the age of eight, has also developed intranet Web pages for a leading telecommunications and networking company in addition to working on several Internet sites. He has programmed in everything from C++, Java, Perl, Visual Basic, and JavaScript to Lingo as well as having experience with both HTML/XHTML and DHTML. He has also published roughly a dozen books on various Internet technologies that include topics such as Perl, JavaScript, PHP, and XML. In the past, he has also written the monthly “Webmaster” column for SunWorld and a weekly article, “Integrating Windows and Unix,” for ITworld.com.
Andrew Watt

Andrew Watt is an independent consultant and author based in the United Kingdom with an interest and expertise in the growing family of XML technologies. He wrote his first programs in 6502 Assembly Language and BBC Basic around 1985 and has programmed in Pascal, Prolog, Lotus Domino, and a variety of Web and other technologies including HTML/XHTML and JavaScript. He works with XML, XSLT, SVG, and various other XML technologies on a regular basis and is excited by the progressive transition of the XML technologies from potential to reality as the pieces of the XML jigsaw puzzle appear one by one from the World Wide Web Consortium (W3C).

Andrew is the author of Designing SVG Web Graphics (published by New Riders) and XPath Essentials (published by Wiley) as well as being co-author or contributing author to XHTML, XML & Java 2 Platinum Edition (published by Que), Professional XSL, Professional XML 2nd Edition and Professional XML Meta Data (published by Wrox), and Sams Teach Yourself JavaScript in 21 Days (in press at Sams).
Getting Started
The World Wide Web Consortium’s XML Schema is arguably one of the most important and far-reaching recommendations related to XML to come from the W3C.

Since its introduction as a W3C recommendation in 1998, *Extensible Markup Language* (XML) has had a rapidly growing impact on the World Wide Web and as a basis for electronic business. As the impact of XML has grown, the need to integrate XML with existing technologies, such as programming languages and relational database management systems, and the need to exchange information expressed in XML has led to demands for a schema language written in XML that will constrain the allowed structure of a class of XML documents with precision and that can also constrain the datatypes that are permitted at individual locations within such a structure. The need for a new schema language arose, in part, from the limitations of the *Document Type Definition* (DTD), which was the form of XML schema defined within the XML 1.0 Recommendation of February 1998.

As well as being one of the most important recommendations, the W3C XML Schema Recommendation is one of the most complex, and at times abstract, XML technology specifications. In this book, we will be emphasizing aspects of W3C XML Schema that are practical, using many examples of W3C
XML schemas and introducing the theory that sheds light on the practical use of schemas.

Let’s take a quick look at a simple XML schema so that you can see what one looks like. An XML document that is described by an XML schema is called an instance document. Listing 1.1 shows a very simple XML instance document.

A schema expressed in W3C XML Schema syntax that describes the permitted content of Listing 1.1 is shown in Listing 1.2. The details of the syntax are not essential for you to understand at this stage.

As you can see, the schema of XML Schema is substantially longer than the document it describes or defines. For the moment, do not worry about the detail of the schema. The <xsd:annotation> and <xsd:documentation> elements enable us to document the purpose of a schema for a human reader. The <xsd:element> and <xsd:attribute> elements enable us to declare elements and attributes that are permitted in instance documents. The <xsd:complexType> element enables us to define the permitted complex type content of certain elements. How to use XSD Schema elements such as <xsd:element>, <xsd:complexType>, <xsd:attribute>, and so on will be introduced a little later in this chapter.

**NOTE** The World Wide Web Consortium, or W3C, has termed its version of a schema language as simply XML Schema. In reality, a number of other XML schema languages existed for some time before W3C completed the development of XML Schema. So, to avoid ambiguity, when we refer to the specification for the W3C flavor of XML Schema, we will use the terms W3C XML Schema or XSD Schema to refer to W3C’s type of XML Schema, because an earlier name for the W3C XML Schema was XML Schema Definition Language, abbreviated to XSD. When we refer to a specific example of a schema written in the XSD Schema language (with the upper-case initial letter of Schema), we will use the term XSD schema (with the lower-case initial letter of schema).

Throughout this book, we will be using the indicative namespace prefix `xsd` to refer to elements such as `<xsd:complexType>` (which are part of XSD Schema).

```
<?xml version="1.0"?>
<Book>
  <Title>XML Schema Essentials</Title>
  <Authors>
    <Author>R. Allen Wyke</Author>
    <Author>Andrew Watt</Author>
  </Authors>
  <Publisher>John Wiley</Publisher>
</Book>
```

**Listing 1.1** Simple XML instance document (Book.xml).
<xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

<xsd:annotation>
  <xsd:documentation>
    This is a sample XML Schema for Chapter 1 of XML Schema Essentials.
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" ref="Title"/>
      <xsd:element name="Authors" ref="Authors"/>
      <xsd:element name="Publisher" ref="Publisher"/>
    </xsd:sequence>
    <xsd:attribute name="pubCountry" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Title" type="xsd:string"/>

<xsd:element name="Authors">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Author" ref="Author" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Author" type="xsd:string"/>

<xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>

**Listing 1.2**  W3C XML Schema syntax describing content of Listing 1.1 (Book.xsd).

**What Is XML Schema?**

XML Schema is the W3C-recommended schema definition language, expressed in XML 1.0 syntax, which is intended to describe the structure and constrain the content of documents written in XML. It is explicitly intended to improve on the schema functionality that was provided by the DTD, which was the original form of schema for XML documents that the W3C recommended in 1998 when XML was first released.
The W3C XML Schema became a full W3C recommendation in May 2001. Unusually, the final recommendation was released in three parts. The first part, Part 0, is a primer that is intended to introduce XML Schema in a nonformal way (from W3C’s point of view) and is located at www.w3.org/TR/2001/REC-xmlschema-0-20010502/. Part 1 is a normative W3C document, defines structures that XML Schema supports, and is located at www.w3.org/TR/2001/REC-xmlschema-1-20010502/. Part 2 is also a normative W3C document, defines the datatypes that W3C XML Schema supports, describes mechanisms for creating new datatypes, and is located at www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

An XSD Schema schema is intended to define the structure and constrain the content of a class of XML documents. Given the terminology “class,” such documents are often termed instance documents.

**NOTE** Instance “documents” need not exist as document files but can exist as a stream of bytes or as a collection of XML Information Set items.

How Does an XML Schema Processor Work?

In much of this book, we will refer to the relationship between an XSD schema and instance documents as if an XSD schema-aware validating processor actually directly processed the instance document. In fact, an XSD schema-aware processor operates on a set (called the information set) of information items rather than on the instance document itself. This method is similar to the way that an XSLT/XPath processor operates, in reality, on a source tree of nodes rather than directly on the elements in a source XML document. Later in this chapter, we will take a look at the XML Information Set specification and examine how the XML Information Set is relevant to XSD Schema.

It isn’t surprising that an XSD Schema processor does not operate directly on an XML instance document; after all, an instance document is simply a series of characters. An XML parser extracts a series of logical tokens by parsing the characters in the serialized document. In the case of a parser that is XML Information Set-aware, the logical tokens are termed information items. There is, for example, a document information item (broadly corresponding to the document entity) that has several properties. Among the properties of the document information item is the [children] property. Note that the name of a property of an information item, such as the [children] property, is written enclosed in square brackets. One of the information items in the [children] property of the document information item is the element information item, which represents the document element of the instance document.
What Is XML Schema for?

The purpose of XML Schema is to define the structure of XML instance documents. By defining and constraining the content of XML instance documents, it becomes possible to exchange information between applications with greater confidence and with less custom programming to test and confirm the structure of an instance document, or to confirm that the data in a particular part of the document is of a particular datatype.

XSD Schema adds the capability to combine schemas from more than one source. For example, we could generate an invoice perhaps by combining a schema from a customer’s purchase order (which includes information such as shipping address, billing address, and so on) and billing information from our own accounts department (describing information such as price, discount allowed, and so on). This technique would enable schemas to be reused in a variety of combinations, thus improving efficiency.

XSD Schema Schema Components

The W3C XML Schema Recommendation indicates that an XSD schema comprises 13 types of schema components that fall broadly into three groups: primary, secondary, and helper components.

The XSD Schema Recommendation refers to the following primary components:

- Simple type definitions
- Complex type definitions
- Attribute declarations
- Element declarations

Primary components that are type definitions can have names. Attribute declarations and element declarations must have names.

The following are the secondary components:

- Attribute group definitions
- Identity-constraint definitions
- Model group definitions
- Notation declarations

The final five XSD Schema components are referred to as helper components and provide parts of other components:

- Annotations
- Model groups
Chapter 1

• Particles
• Wildcards
• Attribute uses

This chapter introduces the syntax to enable you to use many of the components just mentioned. Later chapters will detail how they are to be used.

Other Schema Languages for XML

Other schema languages are written in XML and are designed for use in defining and describing XML instance documents. This book does not describe them in detail because that is not its intended purpose. You should be aware of the existence of these other schema languages, however, and where you can obtain information about them.

XML-Data Reduced, often known simply as XDR, is a schema language that antedated the XSD Schema language. XDR is routinely used within the BizTalk Framework (www.biztalk.org) sponsored by Microsoft and is supported by Microsoft’s MSXML parser.

Another important alternative schema language for XML is now termed RELAX NG. RELAX NG, standing for RELAX New Generation, is an amalgam of two embryonic schema languages, RELAX and TREX. RELAX NG is being developed by the Organization for Advancement of Structured Information Standards (OASIS), found at www.oasis-open.org.

These XML schemas are written for XML as well as being written in XML. The original schema for XML 1.0 was the DTD that was, however, not written in XML.

The DTD Descended from SGML

The first form of schema for XML documents was the Document Type Definition. Definitive information about the XML Document Type Definition is contained in the XML 1.0 Recommendation. At the time that XML became a recommendation, few people envisaged how it would evolve from being a document description language into one that would be used for many data-centric, rather than document-centric, applications. Not surprisingly, then, the DTD created largely with document-centric use in mind was found to have inadequacies when routinely applied in a data-centric context.

Among the limitations of the DTD are the following:

• Datatyping is very weak.
• DTDs have a limited range of content models.
The content cannot be validated precisely where it is of mixed content type.
Cardinality is limited to being defined to zero, one, or many occurrences.
DTDs lack named element or attribute groups that would enable us to reuse them.
XSD Schema was designed, among other things, to provide superior datatyping to the DTD, to provide greater flexibility but yet with control of content models, and to provide definitions of cardinality not possible using a DTD.

Later in the chapter, we will look in a little more detail at comparisons between DTDs and XSD schemas once you have been introduced to some commonly used XSD Schema constructs.

XSD Schema Tools

You might well already have your own favorite tools with which to create XML documents and XSD schemas. If so, then feel free to use these as you work through the examples in this book. The tools mentioned here include tools that the authors use on an ongoing basis. We are making no specific claims for their superiority for a particular purpose, but they do enable us to work with XSD Schema to explore its capabilities and complexities.

NOTE Each of the XSD Schema tools is an early implementation; therefore, you can expect to find some situations where inappropriate error messages are issued or where an error in a schema is overlooked.

Schema Checkers

At this writing, an online XSD Schema checking service is available using XML Schema Validator provided at the W3C Web site. The schema validation service for the May 2001 Recommendation is located at www.w3.org/2001/03/webdata/xsv. Figure 1.1 shows part of the form that you must use in order to get a schema validated. Essentially, your schema needs to be accessible via a URL in order to be validated.

The online checking service can check an XSD schema for validity (that is, conformity to the W3C Recommendation), or it can validate an instance document against a schema.

The online schema checking service can be used to process files that are accessible at a URL, or you can upload files for checking.
Figure 1.1  Online XSD schema validation service from the W3C.

Figure 1.2 shows the form filled in just prior to asking the processor to validate an instance document (Book.xml, Listing 1.1) against its XSD schema (Book.xsd, Listing 1.2).

Be careful to include the http:// part of the URL; otherwise, the schema checker interprets the URL as being a relative one, as shown in Figure 1.3. Relative URLs are not permitted in XSD Schema.

The output from validating Book.xml, using Book.xsd, is shown in Figure 1.4.

NOTE  XSV is undergoing continuing development. At this writing, not all parts of the W3C Recommendation are supported. The download page mentioned earlier provides details of areas not yet fully implemented.

The W3C schema checker requires that you either make the file(s) available at a URL or upload them by using the online form. If you are unable or unwilling to do that, an alternate approach is to download the schema checker that lies behind the W3C schema checking service.

Two schema checkers are available for download. One from Henry Thomson at the University of Edinburgh (also of the W3C XML Schema Working Group) is the basis of the W3C schema checking service. The other download is available from IBM.
Figure 1.2  Using the online schema checker.

Figure 1.3  Error message if http:// is omitted.

The download version of XML Schema Validator, XSV, corresponds to the online schema checking service at W3C. Further information about downloading XML Schema Validator can be found at www.ltg.ed.ac.uk/~ht/xsv-status.html.

To check the validity of the Book.xsd schema, copy it to the XSV directory (or place the directory containing Book.xsd in the PATH) and issue the following command:

```
xsv -i Book.xsd
```
Figure 1.4  The result from the W3C online schema validator when validation has been successful.

Figure 1.5  Output of validating Book.xsd by using the XSV schema validator.

and you will see output like that in Figure 1.5. Note that there are zero schema errors.
Alternatively, you can issue the command

```
xsv -o output.xml -s xsv.xsl -i Book.xsd
```

And, if you have MSXML3 installed, you will see output similar to the display of the output file output.xml, as shown in Figure 1.6.

The IBM XML Schema Quality Checker can be downloaded from www.alphaworks.ibm.com/tech/xmlsqc. The IBM XML Schema Quality Checker checks whether or not an XSD schema corresponds to the W3C Recommendation. It does not, at least at this writing, validate instance documents against the XSD schema.

As well as standalone schema validation tools such as those just described schemas can be validated by using schema authoring tools.

**Schema Authoring Tools**

XSD Schemas can be created by using any XML editor, but editors that are not XSD Schema-aware are limited as learning and production tools. They can spot basic XML syntax errors and can indicate that the syntax is well-formed, but are incapable of providing information about the correctness or incorrectness of a schema you have created. Therefore, you would be well-advised to consider, if you have not already done so, acquiring a schema editor such as Turbo XML (from TIBCO Extensibility) or XML Spy (from Altova). Both have free evaluation downloads available from their respective Web sites. Turbo XML

![Figure 1.6 Using XSV schema validator and an XSLT stylesheet to generate an output file, output.xml.](image)

Turbo XML is available for various flavors of 32-bit Windows operating systems, multiple flavors of Unix and for Mac OS X. Occasionally version 2.2.1 overlooks schema errors correctly identified by XML Spy.

**NOTE** If you have Turbo XML version 2.2.1 running, you might find that you cannot start the Netscape 6 browser. If you use Netscape 6, start the browser before starting Turbo XML.

The generally available version of XML Spy at this writing is version 4.0. Version 3.5 does not use the final XSD Schema namespace. XML Spy Version 4 supports the full XSD (W3C XML) Schema Recommendation.

XML Spy is generally easy to use. One irritation with XML Spy 4.0 is that it reformats code, however. For example, it introduces tabs instead of spaces. In addition, occasional spurious error messages are produced.

Despite the minor problems just mentioned, both Turbo XML and XML Spy are powerful and useful tools for XSD Schema development. Each is capable of validating an instance document against a schema as well as validating an XSD schema for conformity to the W3C XML Schema Recommendation.

**XML Schema Document**

In this section, we will look briefly at the general structure of an XML Schema schema such as the one that you saw earlier in Listing 1.2. The description of each part of a schema document will be brief, and many points will be developed in greater depth in later chapters.

An XSD Schema document begins, optionally, with an XML declaration with required version attribute and optional encoding and standalone attributes:

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

Then follows the `<xsd:schema>` element, which is the element root of all XSD Schema documents. On the `<xsd:schema>` element, there is a namespace declaration for the XSD Schema namespace:

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
```

**NOTE** If you see an XSD schema where the namespace declaration refers to www.w3.org/2000/10/XMLSchema, that indicates that the schema was created by using a non-final version of the specification.

```xml
<xsd:annotation>
  <xsd:documentation>
```