

# JSR-000206 Java™ API for XML Processing (“Specification”)

*Version: 1.4*

*Maintenance Release*

*16 August 2006*

Editor:  
Norman Walsh

Comments to: [eg@jaxp.dev.java.net](mailto:eg@jaxp.dev.java.net)

*Sun Microsystems, Inc.  
4150 Network Circle  
Santa Clara, CA 95054  
USA*

---

# Specification: JSR-000206 Java™ API for XML Processing (“Specification”)

Version: 1.4

Status: Maintenance Release

Release: 16 August 2006

Copyright © 2006 SUN MICROSYSTEMS, INC.

4150 Network Circle, Santa Clara, California 95054, U.S.A

All rights reserved.

## LIMITED LICENSE GRANTS

1. *License for Evaluation Purposes.* Sun hereby grants you a fully-paid, non-exclusive, non-transferable, worldwide, limited license (without the right to sublicense), under Sun's applicable intellectual property rights to view, download, use and reproduce the Specification only for the purpose of internal evaluation. This includes (i) developing applications intended to run on an implementation of the Specification, provided that such applications do not themselves implement any portion(s) of the Specification, and (ii) discussing the Specification with any third party; and (iii) excerpting brief portions of the Specification in oral or written communications which discuss the Specification provided that such excerpts do not in the aggregate constitute a significant portion of the Specification.

2. *License for the Distribution of Compliant Implementations.* Sun also grants you a perpetual, non-exclusive, non-transferable, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights or, subject to the provisions of subsection 4 below, patent rights it may have covering the Specification to create and/or distribute an Independent Implementation of the Specification that: (a) fully implements the Specification including all its required interfaces and functionality; (b) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and (c) passes the Technology Compatibility Kit (including satisfying the requirements of the applicable TCK Users Guide) for such Specification ("Compliant Implementation"). In addition, the foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose (including, for example, modifying the Specification, other than to the extent of your fair use rights, or distributing the Specification to third parties). Also, no right, title, or interest in or to any trademarks, service marks, or trade names of Sun or Sun's licensors is granted hereunder. Java, and Java-related logos, marks and names are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

3. *Pass-through Conditions.* You need not include limitations (a)-(c) from the previous paragraph or any other particular "pass through" requirements in any license You grant concerning the use of your Independent Implementation or products derived from it. However, except with respect to Independent Implementations (and products derived from them) that satisfy limitations (a)-(c) from the previous paragraph, You may neither: (a) grant or otherwise pass through to your licensees any licenses under Sun's applicable intellectual property rights; nor (b) authorize your licensees to make any claims concerning their implementation's compliance with the Specification in question.

4. *Reciprocity Concerning Patent Licenses.*

a. With respect to any patent claims covered by the license granted under subparagraph 2 above that would be infringed by all technically feasible implementations of the Specification, such license is conditioned upon your offering on fair, reasonable and non-discriminatory terms, to any party seeking it from You, a perpetual, non-exclusive, non-transferable, worldwide license under Your patent rights which are or would be infringed by all technically feasible implementations of the Specification to develop, distribute and use a Compliant Implementation.

b. With respect to any patent claims owned by Sun and covered by the license granted under subparagraph 2, whether or not their infringement can be avoided in a technically feasible manner when implementing the Specification, such license shall terminate with respect to such claims if You initiate a claim against Sun that it has, in the course of performing its responsibilities as the Specification Lead, induced any other entity to infringe Your patent rights.

c. Also with respect to any patent claims owned by Sun and covered by the license granted under subparagraph 2 above, where the infringement of such claims can be avoided in a technically feasible manner when implementing the Specification such license, with respect to such claims, shall terminate if You initiate a claim against Sun that its making, having made, using, offering to sell, selling or importing a Compliant Implementation infringes Your patent rights.

5. *Definitions.* For the purposes of this Agreement: "Independent Implementation" shall mean an implementation of the Specification that neither derives from any of Sun's source code or binary code materials nor, except with an appropriate and separate license from Sun, includes any of Sun's source code or binary code materials; "Licensor Name Space" shall mean the public class or interface declarations whose names begin with "java", "javax", "com.sun" or their equivalents in any subsequent naming convention adopted by Sun through the Java Community Process, or any recognized successors or replacements thereof; and "Technology Compatibility Kit" or "TCK" shall mean the test suite and accompanying TCK User's Guide provided by Sun which corresponds to the Specification and that was available either (i) from Sun's 120 days before the first release of Your Independent Implementation that allows its use for commercial purposes, or (ii) more recently than 120 days from such release but against which You elect to test Your implementation of the Specification.

---

---

This Agreement will terminate immediately without notice from Sun if you breach the Agreement or act outside the scope of the licenses granted above.

## **DISCLAIMER OF WARRANTIES**

THE SPECIFICATION IS PROVIDED "AS IS". SUN MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT (INCLUDING AS A CONSEQUENCE OF ANY PRACTICE OR IMPLEMENTATION OF THE SPECIFICATION), OR THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE. This document does not represent any commitment to release or implement any portion of the Specification in any product. In addition, the Specification could include technical inaccuracies or typographical errors.

## **LIMITATION OF LIABILITY**

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED IN ANY WAY TO YOUR HAVING, IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION, EVEN IF SUN AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend Sun and its licensors from any claims arising or resulting from: (i) your use of the Specification; (ii) the use or distribution of your Java application, applet and/or implementation; and/or (iii) any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

## **RESTRICTED RIGHTS LEGEND**

U.S. Government: If this Specification is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

## **REPORT**

If you provide Sun with any comments or suggestions concerning the Specification ("Feedback"), you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Sun a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose.

## **GENERAL TERMS**

Any action related to this Agreement will be governed by California law and controlling U.S. federal law. The U.N. Convention for the International Sale of Goods and the choice of law rules of any jurisdiction will not apply.

The Specification is subject to U.S. export control laws and may be subject to export or import regulations in other countries. Licensee agrees to comply strictly with all such laws and regulations and acknowledges that it has the responsibility to obtain such licenses to export, re-export or import as may be required after delivery to Licensee.

This Agreement is the parties' entire agreement relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, conditions, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification to this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

Rev. April, 2006

Sun/Final/Full

---

---

# Table of Contents

1. Overview .....	1
What is XML? .....	1
XML and the Java™ Platform .....	1
About This Specification .....	1
Who Should Read This Document .....	1
Report and Contact .....	2
Development of This Specification .....	2
Development of the JAXP 1.3 Specification .....	2
Development of the JAXP 1.4 Specification .....	3
Acknowledgments .....	3
2. Related Technologies .....	5
Streaming API for XML (StAX) .....	5
XML 1.1 Support in StAX .....	5
3. Endorsed Specifications .....	6
Extensible Markup Language (XML) .....	6
Namespaces in XML .....	6
XML Schema .....	6
XSL Transformations (XSLT) .....	7
XML Path Language (XPath) .....	7
XML Inclusions (XInclude) .....	7
Document Object Model (DOM) Level 2 .....	7
Document Object Model (DOM) Level 3 .....	8
Simple API for XML (SAX) .....	8
4. Plugability Layer .....	9
SAX Plugability .....	9
Examples .....	9
DOM Plugability .....	10
Reliance on SAX API .....	11
Examples .....	11
XSLT Plugability .....	12
Examples .....	12
XPath Plugability .....	17
Examples .....	18
Validation Plugability .....	18
Streaming API for XML Plugability .....	19
Examples .....	19
Datatype Plugability .....	20
Thread Safety .....	21
Properties For Enabling Schema Validation .....	21
Samples Using the Properties .....	23
Recommended Implementation of Properties .....	24
5. Conformance Requirements .....	25
6. XML Inclusions (XInclude) .....	26
What is XML Inclusions (XInclude) .....	26
Implementation Required by JSR-000206 Java™ API for XML Processing (“Specification”) .....	26
7. JSR-000206 Java™ API for XML Processing (“Specification”) Specification and Implementation	
Version Information .....	27
JSR-000206 Java™ API for XML Processing (“Specification”) Specification Version Information .....	27

JSR-000206 Java™ API for XML Processing (“Specification”) Implementation Version In- formation .....	28
8. Constants .....	29
9. Changes Since JAXP 1.3 .....	35
Package javax.xml.datatype .....	35
Class DatatypeFactory .....	35
Class Duration .....	35
Class XMLGregorianCalendar .....	35
Package javax.xml.parsers .....	35
Class DocumentBuilderFactory .....	35
Class SAXParserFactory .....	35
Package javax.xml.stream .....	35
Package javax.xml.stream.events .....	36
Package javax.xml.stream.util .....	37
Package javax.xml.transform .....	37
Class ErrorListener .....	37
Class Transformer .....	37
Class TransformerFactory .....	37
Package javax.xml.transform.stax .....	37
Package javax.xml.validation .....	37
Class Schema .....	37
Class SchemaFactory .....	38
Class SchemaFactoryLoader .....	38
Class TypeInfoProvider .....	38
Class Validator .....	38
Package javax.xml.xpath .....	38
Class XPath .....	38
Class XPathFactory .....	39
10. Colophon .....	40
Talk the Talk, Walk the Walk .....	40

---

## List of Tables

7.1. JSR-000206 Java™ API for XML Processing (“Specification”) Specification Version Information .....	27
8.1. Constants in javax.xml.datatype.DatatypeFactory .....	29
8.2. Constants in javax.xml.datatype.DatatypeConstants .....	30
8.3. Constants in javax.xml.transform.dom.DOMResult .....	31
8.4. Constants in javax.xml.transform.dom.DOMSource .....	31
8.5. Constants in javax.xml.transform.sax.SAXResult .....	31
8.6. Constants in javax.xml.transform.sax.SAXSource .....	31
8.7. Constants in javax.xml.transform.sax.SAXTransformerFactory .....	32
8.8. Constants in javax.xml.transform.stream.StreamResult .....	32
8.9. Constants in javax.xml.transform.stream.StreamSource .....	32
8.10. Constants in javax.xml.transform.OutputKeys .....	32
8.11. Constants in javax.xml.transform.Result .....	32
8.12. Constants in javax.xml.transform.stax.StAXResult .....	32
8.13. Constants in javax.xml.transform.stax.StAXSource .....	33
8.14. Constants in javax.xml.xpath.XPathConstants .....	33
8.15. Constants in javax.xml.xpath.XPathFactory .....	33
8.16. Constants in javax.xml.stream.XMLInputFactory .....	33
8.17. Constants in javax.xml.stream.XMLOutputFactory .....	33
8.18. Constants in javax.xml.stream.XMLStreamConstants .....	34
8.19. Constants in javax.xml.XMLConstants .....	34

---

# Chapter 1. Overview

## What is XML?

XML is the meta language defined by the World Wide Web Consortium (W3C) that can be used to describe a broad range of hierarchical mark up languages. It is a set of rules, guidelines, and conventions for describing structured data in a plain text, editable file. Using a text format instead of a binary format allows the programmer or even an end user to look at or utilize the data without relying on the program that produced it. However the primary producer and consumer of XML data is the computer program and not the end-user.

Like HTML, XML makes use of tags and attributes. Tags are words bracketed by the “<” and “>” characters and attributes are strings of the form 'name="value"' that are inside of tags. While HTML specifies what each tag and attribute means, as well as their presentation attributes in a browser, XML uses tags only to delimit pieces of data and leaves the interpretation of the data to the application that uses it. In other words, XML defines only the structure of the document and does not define any of the presentation semantics of that document.

Development of XML started in 1996 leading to a W3C Recommendation in February of 1998. However, the technology is not entirely new. It is based on SGML (Standard Generalized Markup Language) which was developed in the early 1980's and became an ISO standard in 1986. SGML has been widely used for large documentation projects and there is a large community that has experience working with SGML. The designers of XML took the best parts of SGML, used their experience as a guide and produced a technology that is just as powerful as SGML, but much simpler and easier to use.

XML-based documents can be used in a wide variety of applications including vertical markets, e-commerce, business-to-business communication, and enterprise application messaging.

## XML and the Java™ Platform

In many ways, XML and the Java Platform form an ideal partnership. XML defines a cross platform data format and Java provides a standard cross platform programming platform. Together, XML and Java technologies allow programmers to apply Write Once, Run Anywhere™ fundamentals to the processing of data and documents generated by both Java based programs and non-Java based programs.

## About This Specification

This document describes the Java API for XML Processing, Version 1.4. This version of the specification introduces basic support for parsing and manipulating XML documents through a standardized set of Java Platform APIs.

When this specification is final there will be a Reference Implementation which will demonstrate the capabilities of this API and will provide an operational definition of the specification. A Technology Compatibility Kit (TCK) will also be available that will verify whether an implementation of this specification is compliant. These are required as per the Java Community Process 2.5 (JCP 2.5).

## Who Should Read This Document

This specification is intended for use by:

- Parser Developers wishing to implement this version of the specification in their parser.
- Application Developers who use the APIs described in this specification and wish to have a more complete understanding of the API.
- Implementors of XPath and XSLT who wish to support the APIs in this specification, particularly the Transformation and XPath APIs.

This specification is not a tutorial or a user's guide to XML, DOM, SAX, StAX or XSLT. Familiarity with these technologies and specifications on the part of the reader is assumed.

## Report and Contact

Your comments on this specification are welcome and appreciated. Without your comments, the specifications developed under the auspices of the Java Community Process would not serve your needs as well. To comment on this specification, please send email to `<eg@jaxp.dev.java.net>`.

You can stay current with Sun's Java Platform related activities, as well as information on our `<xml-interest>` and `<xml-announce>` mailing lists, at our website [<http://java.sun.com/xml/jaxp>].

## Development of This Specification

This is a maintenance release of JAXP 1.3.

### Development of the JAXP 1.3 Specification

The JAXP 1.3 specification was developed in accordance with the Java Community Process [<http://www.jcp.org>] 2.5. It was developed under the authorization of Java Specification Request 206 [<http://jcp.org/en/jsr/detail?id=206>].

The expert group who contributed to JAXP 1.3 was composed of individuals from a number of companies:

- Ben Galbraith
- Neil Graham, IBM
- Phil Hanna, SAS Institute Inc.
- Todd Karakashian, BEA
- K. Karun, Oracle
- Kohsuke Kawaguchi, Sun Microsystems, Inc.
- Dongeun Kim, Tmax Soft, Inc.
- Harish Krishnaswamy, Novell, Inc.
- Miles Sabin
- Vladimir Savchenko, SAP AG
- Ilene Seelemann, IBM



- Jeff Suttor, Sun Microsystems, Inc.
- Norman Walsh (Specification Lead), Sun Microsystems, Inc.
- Henry Zongaro, IBM

We would like to acknowledge that a lot of the grammar caching work introduced in this version of the specification was derived from the work being done under the Xerces project at Apache.

## Development of the JAXP 1.4 Specification

The informal expert group who contributed to this maintenance release is composed of individuals from a number of companies. These individuals are:

- Neeraj Bajaj, Sun Microsystems, Inc.
- Ben Galbraith
- Michael Glavashevich, IBM
- K. Karun, Oracle
- Kohsuke Kawaguchi, Sun Microsystems, Inc.
- Michael Kay, Saxonica
- Suresh Kumar, Sun Microsystems, Inc.
- Santiago Pericas-Geersten (Specification Lead), Sun Microsystems, Inc.
- Jeff Suttor, Sun Microsystems, Inc.
- Norman Walsh (Specification Lead), Sun Microsystems, Inc.

## Acknowledgments

Many individuals and companies have given their time and talents to make this specification, or the specifications that this specification relies upon, a reality. The authors of this specification would like to thank (in no particular order):

- David Brownell, David Megginson and the XML-DEV community who developed the SAX API
- The W3C DOM Working Group chaired by Philippe Le Hégaré
- The Apache Software Foundation [<http://apache.org/>], for Xerces and Xalan
- Michael Kay, Saxonica [<http://www.saxonica.com/>]
- Elliotte Rusty Harold, Cafe con Leche XML News and Resources [<http://www.cafeconleche.org/>], Cafe au Lait Java News and Resources [<http://www.cafeaulait.org/>]
- Han Ming Ong, Apple
- Eduardo Pelegri-Lopart, Tom Kincaid, Connie Weiss, Gopal Sharma, Neeraj Bajaj, K. Venugopal, Arun Yadav, Ramesh Mandava, Bhakti Mehta, Prasad Subramanian, Todd Miller, Joesph Fialli, and Rajiv

Mordani all of whom work at Sun Microsystems, Inc. and whose talents have all reflected upon the development of this API.

- Margaret L. Wade for allowing it all to be true.

---

# Chapter 2. Related Technologies

This specification builds upon other technologies developed within the Java Community Process [<http://jcp.org/>]. Each technology used by this document is called out together with the exact version of the specification and its publicly accessible location

## Streaming API for XML (StAX)

This specification supports Streaming API for XML (StAX) [<http://jcp.org/en/jsr/detail?id=173>].

This specification includes by reference *Streaming API for XML (StAX) Version 1.0* (JSR 173) in its entirety for the purposes of defining Streaming API for XML (StAX) in the APIs defined herein.

The API packages included by reference are:

- `javax.xml.stream`
- `javax.xml.stream.events`
- `javax.xml.stream.util`

## XML 1.1 Support in StAX

The XML 1.1 Recommendation was published after the StAX APIs. As a result, the StAX APIs make no reference to XML 1.1. However, the JAXP API, since version 1.3, has supported XML 1.1.

In order to assure the widest possible interoperability and the least amount of user confusion, JAXP 1.4 imposes the additional requirement that the StAX APIs used in a JAXP context must support XML 1.1. We hope and expect that a future version of StAX will officially endorse XML 1.1 and make this additional requirement redundant.

---

# Chapter 3. Endorsed Specifications

This specification endorses and builds upon several external specifications. Each specification endorsed by this document is called out together with the exact version of the specification and its publicly accessible location. All of these standards have conformance tests provided in the Technology Compatibility Kit available for this specification.

## Extensible Markup Language (XML)

This specification supports Extensible Markup Language (XML) 1.1 [<http://www.w3.org/TR/xml11>] (and XML 1.1 First Edition Specification Errata [<http://www.w3.org/XML/xml-V11-1e-errata>]) and Extensible Markup Language (XML) 1.0 (Third Edition) [<http://www.w3.org/TR/REC-xml>] (and Extensible Markup Language (XML) 1.0 (Third Edition) Errata [<http://www.w3.org/XML/xml-V10-3e-errata>]).

XML is a product of the W3C XML Activity [<http://www.w3.org/XML/>].

This specification includes by reference Extensible Markup Language (XML) 1.1, XML 1.1 First Edition Specification Errata, Extensible Markup Language (XML) 1.0 (Third Edition) and Extensible Markup Language (XML) 1.0 (Third Edition) Errata in their entirety for the purposes of defining XML in the APIs defined herein.

### A Note about XML Versions

XML 1.0 and XML 1.1 are not completely interchangeable. Developers working in environments where a mixture of versions may exist must consider carefully how serialization and transformation may interact with XML versions.

## Namespaces in XML

This specification supports Namespaces in XML 1.1 [<http://www.w3.org/TR/xml-names11/>] (and Namespaces in XML 1.1 Errata [<http://www.w3.org/XML/2004/xml-names11-errata>]) and Namespaces in XML 1.0 [<http://www.w3.org/TR/REC-xml-names>] (and Namespaces in XML 1.0 Errata [<http://www.w3.org/XML/xml-names-19990114-errata>]).

Namespaces in XML is a product of the W3C XML Activity [<http://www.w3.org/XML/>].

This specification includes by reference Namespaces in XML 1.1, Namespaces in XML 1.1 Errata, Namespaces in XML 1.0 and Namespaces in XML 1.0 Errata, in their entirety for the purposes of defining Namespaces in XML in the APIs defined herein.

## XML Schema

This specification supports XML Schema Part 1: Structures Second Edition [<http://www.w3.org/TR/xmlschema-1/>], XML Schema Part 1: Structures Second Edition Errata [<http://www.w3.org/2004/03/xmlschema-errata#Errata1>], XML Schema Part 2: Datatypes Second Edition [<http://www.w3.org/TR/xmlschema-2/>] and XML Schema Part 2: Datatypes Second Edition Errata [<http://www.w3.org/2004/03/xmlschema-errata#Errata2>].

XML Schema is a product of the XML Schema Working Group [<http://www.w3.org/XML/Schema>].

This specification includes by reference XML Schema Part 1: Structures Second Edition, XML Schema Part 1: Structures Second Edition Errata, XML Schema Part 2: Datatypes Second Edition and XML Schema Part 2: Datatypes Second Edition Errata in their entirety for the purposes of defining XML Schema in the APIs defined herein.

## XSL Transformations (XSLT)

This specification supports XSL Transformations (XSLT) Version 1.0 [<http://www.w3.org/TR/xslt>] and XSL Transformations (XSLT) Version 1.0 Errata [<http://www.w3.org/1999/11/REC-xslt-19991116-errata>].

XSLT is a product of the W3C Style Activity [<http://www.w3.org/Style/Activity>].

This specification includes by reference XSL Transformations (XSLT) Version 1.0 in its entirety for the purposes of defining XSLT in the APIs defined herein.

## XML Path Language (XPath)

This specification supports XML Path Language (XPath) Version 1.0 [<http://www.w3.org/TR/xpath>] and XML Path Language (XPath) Version 1.0 Errata [<http://www.w3.org/1999/11/REC-xpath-19991116-errata>].

XPath is a product of the W3C XML Activity [<http://www.w3.org/XML/Activity>] and W3C Style Activity [<http://www.w3.org/Style/Activity>].

This specification includes by reference XML Path Language (XPath) Version 1.0 and XML Path Language (XPath) Version 1.0 Errata in their entirety for the purposes of defining SAX in the APIs defined herein.

## XML Inclusions (XInclude)

This specification supports XML Inclusions (XInclude) Version 1.0 [<http://www.w3.org/TR/xinclude/>].

XInclude is a product of the W3C XML Core Working Group [<http://www.w3.org/XML/Core/>] as part of the W3C XML Activity [<http://www.w3.org/XML/Activity>].

This specification includes by reference XML Inclusions (XInclude) Version 1.0. in its entirety for the purposes of defining XInclude in the APIs defined herein.

## Document Object Model (DOM) Level 2

This specification supports Document Object Model (DOM) Level 2 Core [<http://www.w3.org/TR/DOM-Level-2-Core>], Document Object Model (DOM) Level 2 Traversal and Range [<http://www.w3.org/TR/DOM-Level-2-Traversal-Range>], and Document Object Model (DOM) Level 2 Events [<http://www.w3.org/TR/DOM-Level-2-Events>].

DOM Level 2 is a product of the W3C DOM Activity [<http://www.w3.org/DOM/>].

This specification includes by reference Document Object Model (DOM) Level 2 Core, Document Object Model (DOM) Level 2 Traversal and Range, and Document Object Model (DOM) Level 2 Events in their entirety for the purposes of defining DOM in the APIs defined herein.

The API packages included by reference are:

- `org.w3c.dom`

- `org.w3c.dom.ranges`
- `org.w3c.dom.traversal`

## Document Object Model (DOM) Level 3

This specification supports Document Object Model (DOM) Level 3 Core [<http://www.w3.org/TR/DOM-Level-3-Core>] and Document Object Model (DOM) Level 3 Load and Save [<http://www.w3.org/TR/DOM-Level-3-LS>].

DOM Level 3 is a product of the W3C DOM Activity [<http://www.w3.org/DOM/>].

This specification includes by reference Document Object Model (DOM) Level 3 Core and Document Object Model (DOM) Level 3 Load and Save in their entirety for the purposes of defining DOM in the APIs defined herein.

The API packages included by reference are:

- `org.w3c.dom`
- `org.w3c.dom.bootstrap`
- `org.w3c.dom.events`
- `org.w3c.dom.ls`

## Simple API for XML (SAX)

This specification supports Simple API for XML (SAX) 2.0.2 (sax2r3) [<http://sax.sourceforge.net/>] and Simple API for XML (SAX) 2.0.2 (sax2r3) Extensions [<http://sax.sourceforge.net/?selected=ext>].

Simple API for XML (SAX) 2.0.2 (sax2r3) is a product of the SAX Community [<http://sax.sourceforge.net/>].

This specification includes by reference Simple API for XML (SAX) 2.0.2 (sax2r3) and Simple API for XML (SAX) 2.0.2 (sax2r3) Extensions in their entirety for the purposes of defining Simple API for XML (SAX) 2.0.2 (sax2r3) in the APIs defined herein.

The API packages included by reference are:

- `org.xml.sax`
- `org.xml.sax.ext`
- `org.xml.sax.helpers`

---

# Chapter 4. Plugability Layer

The endorsed APIs provide broad and useful functionality. However, using the endorsed APIs often requires knowledge of the specific implementations available. Providing the functionality of the endorsed APIs in the Java Platform, while allowing choice of the implementation of the parser, requires a Plugability layer.

This section of the specification defines a Plugability mechanism to allow any compliant implementations to be used through the APIs defined in this specification.

## SAX Plugability

The SAX Plugability classes allow an application programmer to provide an implementation of the `org.xml.sax.DefaultHandler` API to a `SAXParser` implementation and parse XML documents. As the parser processes the XML document, it will call methods on the provided `DefaultHandler`.

In order to obtain a `SAXParser` instance, an application programmer first obtains an instance of a `SAXParserFactory`. The `SAXParserFactory` instance is obtained via one of the static `newInstance` methods of the `SAXParserFactory` class.

This method uses the following ordered lookup procedure to determine the `SAXParserFactory` implementation class to load:

- Use the `javax.xml.parsers.SAXParserFactory` system property.
- Use the properties file "lib/jaxp.properties" in the JRE directory. This configuration file is in standard `java.util.Properties` format and contains the fully qualified name of the implementation class with the key being the system property defined above. The `jaxp.properties` file is read only once by the JSR-000206 Java™ API for XML Processing ("Specification") implementation and its values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in `jaxp.properties` after it has been read for the first time.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file `META-INF/services/javax.xml.parsers.SAXParserFactory` in jars available to the runtime.
- Platform default `SAXParserFactory` instance.

If the `SAXParserFactory` implementation class cannot be loaded or instantiated at runtime, a `FactoryConfigurationError` is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

The instance of `SAXParserFactory` can optionally be configured by the application programmer to provide parsers that are namespace aware, or validating, or both. These settings are made using the `setNamespaceAware` and `setValidating` methods of the factory. The application programmer can then obtain a `SAXParser` implementation instance from the factory. If the factory cannot provide a parser configured as set by the application programmer, then a `ParserConfigurationException` is thrown.

## Examples

The following is a simple example of how to parse XML content from a URL:

```
SAXParser parser;
DefaultHandler handler = new MyApplicationParseHandler();
SAXParserFactory factory = SAXParserFactory.newInstance();
try {
    parser = factory.newSAXParser();
    parser.parse("http://myserver/mycontent.xml", handler);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

The following is an example of how to configure a SAX parser to be namespace aware and validating:

```
SAXParser parser;
DefaultHandler handler = new MyApplicationParseHandler();
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setNamespaceAware(true);
factory.setValidating(true);
try {
    parser = factory.newSAXParser();
    parser.parse("http://myserver/mycontent.xml", handler);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

An example of how one could pass the System property as a command line option is:

```
java -Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl  
user.parserApp
```

## DOM Plugability

The DOM plugability classes allow a programmer to parse an XML document and obtain an `org.w3c.dom.Document` object from a `DocumentBuilder` implementation which wraps an underlying DOM implementation.

In order to obtain a `DocumentBuilder` instance, an application programmer first obtains an instance of a `DocumentBuilderFactory`. The `DocumentBuilderFactory` instance is obtained via one of the static `newInstance` methods of the `DocumentBuilderFactory` class.

This method uses the following ordered lookup procedure to determine the `DocumentBuilderFactory` implementation class to load:

- Use the `javax.xml.parsers.DocumentBuilderFactory` system property
- Use the properties file "lib/jaxp.properties" in the JRE directory. This configuration file is in standard `java.util.Properties` format and contains the fully qualified name of the implementation class with the



key being the system property defined above. The `jaxp.properties` file is read only once by the JSR-000206 Java™ API for XML Processing (“Specification”) implementation and its values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in `jaxp.properties` after it has been read for the first time.

- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file `META-INF/services/javax.xml.parsers.DocumentBuilderFactory` in jars available to the runtime.
- Platform default `DocumentBuilderFactory` instance.

If the `DocumentBuilderFactory` implementation class cannot be loaded or instantiated at runtime, a `FactoryConfigurationError` is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

The instance of `DocumentBuilderFactory` can optionally be configured by the application programmer to provide parsers that are namespace aware or validating, or both. These settings are made using the `setNamespaceAware` and `setValidating` methods of the factory. The application programmer can then obtain a `DocumentBuilder` implementation instance from the factory. If the factory cannot provide a parser configured as set by the application programmer, then a `ParserConfigurationException` is thrown.

## Reliance on SAX API

The `DocumentBuilder` reuses several classes from the SAX API. This does not mean that the implementor of the underlying DOM implementation must use a SAX parser to parse the XML content, only that the implementation communicate with the application using these existing and defined APIs.

## Examples

The following is a simple example of how to parse XML content from a URL:

```
DocumentBuilder builder;
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
String location = "http://myserver/mycontent.xml";
try {
    builder = factory.newDocumentBuilder();
    Document document = builder.parse(location);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

The following is an example of how to configure a factory to produce parsers to be namespace aware and validating:

```
DocumentBuilder builder;
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
factory.setValidating(true);
```

```
String location = "http://myserver/mycontent.xml";
try {
    builder = factory.newDocumentBuilder();
    Document document = builder.parse(location);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

An example of how one could pass the System property as a command line option is:

```
java -Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFact-  
oryImpl user.parserApp
```

## XSLT Plugability

The XSLT Plugability classes allow an application programmer to obtain a Transformer object that is based on a specific XSLT stylesheet from a TransformerFactory implementation. In order to obtain a Transformer object, a programmer first obtains an instance of the TransformerFactory. The TransformerFactory instance is obtained via one of the static newInstance methods of the TransformerFactory class.

This method uses the following ordered lookup procedure to determine the TransformerFactory implementation class to load:

- Use the javax.xml.transform.TransformerFactory system property
- Use the properties file "lib/jaxp.properties" in the JRE directory. This configuration file is in standard java.util.Properties format and contains the fully qualified name of the implementation class with the key being the system property defined above. The jaxp.properties file is read only once by the JSR-000206 Java™ API for XML Processing ("Specification") implementation and its values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in jaxp.properties after it has been read for the first time.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file META-INF/services/javax.xml.transform.TransformerFactory in jars available to the runtime.
- Platform default TransformerFactory instance.

If the TransformerFactory implementation class cannot be loaded or instantiated at runtime, a TransformerFactoryConfigurationError is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

## Examples

The following is a simple example of how to transform XML content:

```
Transformer transformer;
TransformerFactory factory = TransformerFactory.newInstance();
String stylesheet = "file:///home/user/mystylesheet.xsl";
```

```
String sourceId = "file:///home/user/sourcefile.xml";
try {
    transformer = factory.newTransformer(new StreamSource(stylesheet));
    transformer.transform(new StreamSource(sourceId), new StreamResult(System.out))
} catch (Exception e) {
    // handle error
}
```

The following example illustrates the serialization of a DOM node to an XML stream:

```
TransformerFactory tfactory = TransformerFactory.newInstance();
Transformer serializer = tfactory.newTransformer();

Properties oprops = new Properties();
oprops.put("method", "html");
oprops.put("indent-amount", "2");
serializer.setOutputProperties(oprops);

serializer.transform(new DOMSource(doc), new StreamResult(System.out));
```

#### Exceptions and Error Reporting

The following example illustrates the use of the URIResolver to resolve URIs to DOM nodes, in a transformation whose input is totally DOM based:

```
TransformerFactory tfactory = TransformerFactory.newInstance();
if (tfactory.getFeature(DOMSource.FEATURE)
    && tfactory.getFeature(StreamResult.FEATURE)) {
    DocumentBuilderFactory dfactory = DocumentBuilderFactory.newInstance();
    dfactory.setNamespaceAware(true); // Always, required for XSLT
    DocumentBuilder docBuilder = dfactory.newDocumentBuilder();
    // Set up to resolve URLs that correspond to our incl.xml,
    // to a DOM node. Use an anonymous class for the URI resolver.
    final Node xslInc1 = docBuilder.parse("xsl/inc1/inc1.xml");
    final Node xslInc2 = docBuilder.parse("xsl/inc2/inc2.xml");
    tfactory.setURIResolver(new URIResolver() {
        public Source resolve(String href, String base)
            throws TransformerException {
            // ignore base.
            return (href.equals("inc1/inc1.xml"))
                ? new DOMSource(xslInc1)
                : (href.equals("inc2/inc2.xml"))
                ? new DOMSource(xslInc2)
                : null;
        }
    });

    // The TransformerFactory will call the anonymous URI
    // resolver set above when it encounters
    // <xsl:include href="inc1/inc1.xml"/>
    Templates templates
        = tfactory.newTemplates(new DOMSource(docBuilder.parse(xslID), xslID));
```

```
// Get a transformer from the templates.
Transformer transformer = templates.newTransformer();

// Set up to resolve URLs that correspond to our foo2.xml, to
// a DOM node. Use an anonymous class for the URI resolver.
// Be sure to return the same DOM tree every time for the given URI.
final Node xmlSubdir1Foo2Node = docBuilder.parse("xml/subdir1/foo2.xml");
transformer.setURIResolver(new URIResolver() {
    public Source resolve(String href, String base)
        throws TransformerException {
        // ignore base because we're lazy, or we don't care.
        return (href.equals("subdir1/foo2.xml"))
            ? new DOMSource(xmlSubdir1Foo2Node)
            : null;
    }
});

// Now the transformer will call our anonymous URI resolver
// when it encounters the document("subdir1/foo2.xml") invocation.
transformer.transform(new DOMSource(docBuilder.parse(sourceID), sourceID),
    new StreamResult(System.out));
}
```

The following example performs a transformation using DOM nodes as input for the TransformerFactory, as input for the Transformer, and as the output of the transformation:

```
TransformerFactory tfactory = TransformerFactory.newInstance();
// Make sure the TransformerFactory supports the DOM feature.
if (tfactory.getFeature(DOMSource.FEATURE)
    && tfactory.getFeature(DOMResult.FEATURE)) {
    // Use javax.xml.parsers to create our DOMs.
    DocumentBuilderFactory dfactory = DocumentBuilderFactory.newInstance();
    dfactory.setNamespaceAware(true); // do this always for XSLT
    DocumentBuilder docBuilder = dfactory.newDocumentBuilder();

    // Create the Templates from a DOM.
    Node xslDOM = docBuilder.parse(xslID);
    DOMSource dsource = new DOMSource(xslDOM, xslID);
    Templates templates = tfactory.newTemplates(dsource);

    // Create the source tree in the form of a DOM.
    Node sourceNode = docBuilder.parse(sourceID);

    // Create a DOMResult that the transformation will fill in.
    DOMResult dresult = new DOMResult();

    // And transform from the source DOM tree to a result DOM tree.
    Transformer transformer = templates.newTransformer();
    transformer.transform(new DOMSource(sourceNode, sourceID), dresult);

    // The root of the result tree may now be obtained from
    // the DOMResult object.
    Node out = dresult.getNode();
}
```

```

        // Serialize it to System.out for diagnostics.
        Transformer serializer = tfactory.newTransformer();
        serializer.transform(new DOMSource(out), new StreamResult(System.out));
    }

```

The following code fragment illustrates the use of the SAXSource and SAXResult objects:

```

TransformerFactory tfactory = TransformerFactory.newInstance();
// Does this factory support SAX features?
if (tfactory.getFeature(SAXSource.FEATURE)
    && tfactory.getFeature(SAXResult.FEATURE)) {
    // Get a transformer.
    Transformer transformer = tfactory.newTransformer(new StreamSource(xslID));
    // Create a reader for reading.
    XMLReader reader = XMLReaderFactory.createXMLReader();
    transformer.transform(new SAXSource(reader, new InputSource(sourceID)),
        new SAXResult(new ExampleContentHandler()));
}

```

The following illustrates the feeding of SAX events from an `org.xml.sax.XMLReader` to a Transformer:

```

TransformerFactory tfactory = TransformerFactory.newInstance();
// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE)) {
    // If so, we can safely cast.
    SAXTransformerFactory stfactory = ((SAXTransformerFactory) tfactory);
    // A TransformerHandler is a ContentHandler that will listen for
    // SAX events, and transform them to the result.
    TransformerHandler handler
        = stfactory.newTransformerHandler(new StreamSource(xslID));

    // Set the result handling to be a serialization to System.out.
    handler.setResult(new StreamResult(System.out));
    handler.getTransformer().setParameter("a-param", "hello to you!");

    // Create a reader, and set its content handler to be the TransformerHandler.
    XMLReader reader = XMLReaderFactory.createXMLReader();
    reader.setContentHandler(handler);

    // It's a good idea for the parser to send lexical events.
    // The TransformerHandler is also a LexicalHandler.
    reader.setProperty("http://xml.org/sax/properties/lexical-handler", handler);

    // Parse the source XML, and send the parse events to the TransformerHandler.
    reader.parse(sourceID);
}

```

The following code fragment illustrates the creation of a Templates object from SAX2 events sent from an XMLReader:

```

TransformerFactory tfactory = TransformerFactory.newInstance();
// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE)) {

```

```
// If so, we can safely cast.
SAXTransformerFactory stfactory = ((SAXTransformerFactory) tfactory);
// Have the factory create a special ContentHandler that will
// create a Templates object.
TemplatesHandler handler = stfactory.newTemplatesHandler();
// If you don't do this, the TemplatesHandler won't know how to
// resolve relative URLs.
handler.setSystemId(xslID);

// Create a reader, and set its content handler to be the TemplatesHandler.
XMLReader reader = XMLReaderFactory.createXMLReader();
reader.setContentHandler(handler);

// Parse the source XML, and send the parse events to the TemplatesHandler.
reader.parse(xslID);

// Get the Templates reference from the handler.
Templates templates = handler.getTemplates();

// Ready to transform.
Transformer transformer = templates.newTransformer();
transformer.transform(new StreamSource(sourceID), new StreamResult(System.out))
}
```

The following illustrates several transformations chained together. Each filter points to a parent `org.xml.sax.XMLReader`, and the final transformation is caused by invoking `org.xml.sax.XMLReader#parse` on the final reader in the chain:

```
TransformerFactory tfactory = TransformerFactory.newInstance();
// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE)) {
    Templates stylesheet1 = tfactory.newTemplates(new StreamSource(xslID_1));
    Transformer transformer1 = stylesheet1.newTransformer();
    SAXTransformerFactory stf = (SAXTransformerFactory)tfactory;
    XMLReader reader = XMLReaderFactory.createXMLReader();
    XMLFilter filter1 = stf.newXMLFilter(new StreamSource(xslID_1));
    XMLFilter filter2 = stf.newXMLFilter(new StreamSource(xslID_2));
    XMLFilter filter3 = stf.newXMLFilter(new StreamSource(xslID_3));

    // transformer1 will use a SAX parser as its reader.
    filter1.setParent(reader);

    // transformer2 will use transformer1 as its reader.
    filter2.setParent(filter1);

    // transform3 will use transform2 as its reader.
    filter3.setParent(filter2);
    filter3.setContentHandler(new ExampleContentHandler());
    // filter3.setContentHandler(new org.xml.sax.helpers.DefaultHandler());

    // Now, when you call transform3 to parse, it will set
    // itself as the ContentHandler for transform2, and
    // call transform2.parse, which will set itself as the
    // content handler for transform1, and call transform1.parse,
```

```
// which will set itself as the content listener for the
// SAX parser, and call parser.parse(new InputSource("xml/foo.xml")).

filter3.parse(new InputSource(sourceID));
}
```

The following code fragment illustrates the use of the stream Source and Result objects:

```
// Create a TransformerFactory instance.
TransformerFactory tfactory = TransformerFactory.newInstance();

InputStream xslIS = new BufferedInputStream(new FileInputStream(xslID));
StreamSource xslSource = new StreamSource(xslIS);
// Note that if we don't do this, relative URLs cannot be resolved correctly!
xslSource.setSystemId(xslID);

// Create a transformer for the stylesheet.
Transformer transformer = tfactory.newTransformer(xslSource);
InputStream xmlIS = new BufferedInputStream(new FileInputStream(sourceID));
StreamSource xmlSource = new StreamSource(xmlIS);

// Note that if we don't do this, relative URLs cannot be resolved correctly!
xmlSource.setSystemId(sourceID);

// Transform the source XML to System.out.
transformer.transform( xmlSource, new StreamResult(System.out));
```

An example of how one could pass the System property as a command line option is:

```
java -Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactory-
Impl user.parserApp
```

## XPath Plugability

The XPath Plugability classes allow an application programmer to obtain an XPath object that is based on a specific object model implementation. In order to obtain an XPath object, a programmer first obtains an instance of the XPathFactory. The XPathFactory instance is obtained via one of the static newInstance methods of the XPathFactory class.

This method uses the following ordered lookup procedure to determine the XPathFactory implementation class to load:

- Use the `javax.xml.xpath.XPathFactory` system property
- Use the properties file "lib/jaxp.properties" in the JRE directory. This configuration file is in standard `java.util.Properties` format and contains the fully qualified name of the implementation class with the key being the system property defined above. The `jaxp.properties` file is read only once by the JSR-000206 Java™ API for XML Processing ("Specification") implementation and its values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in `jaxp.properties` after it has been read for the first time.

- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file META-INF/services/javax.xml.xpath.XPathFactory in jars available to the runtime.
- Platform default XPathFactory instance.

If the XPathFactory implementation class cannot be loaded or instantiated at runtime, a XPathFactoryConfigurationError is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

## Examples

The following example loads a document and evaluates the XPath expression “/widgets/widget[@name='a']/@quantity” against it.

```
// parse the XML as a W3C Document
DocumentBuilder builder
    = DocumentBuilderFactory.newInstance().newDocumentBuilder();
org.w3c.Document document = builder.parse(new File("/widgets.xml"));

// evaluate the XPath expression against the Document
XPath xpath = XPathFactory.newInstance().newXPath();
String expression = "/widgets/widget[@name='a']/@quantity";
Double quantity = (Double) xpath.evaluate(expression, document,
                                           XPathConstants.NUMBER);
```

The evaluation of XPath expressions can return their results as one of five types: Node, NodeList, Boolean, Double, and String. The third argument to evaluate selects the result type. To get the quantity attribute as a string, use:

```
String str_quantity = (String) xpath.evaluate(expression, document,
                                              XPathConstants.STRING);
```

This may seem somewhat clumsy, but it is necessary because the result type is not a property of the expression, it is a property of the context in which it is evaluated.

## Validation Plugability

The Validation Plugability classes allow an application programmer to obtain a Schema Object that is based on a specific schema language implementation. In order to obtain a Schema Object, a programmer first obtains an instance of the SchemaFactory. The SchemaFactory instance is obtained via one of the static newInstance methods of the SchemaFactory Class.

This method uses the following ordered lookup procedure to determine the SchemaFactory implementation Class to load:

- If the system property javax.xml.validation.SchemaFactory:schemaLanguage is present (where schemaLanguage is the parameter to newInstance), then its value is read as a Class name. The method will try to create a new instance of this Class by using the ClassLoader, and returns it if it is successfully created.
- Read the properties file \$java.home/lib/jaxp.properties in the JRE directory. This configuration file is in standard java.util.Properties format. \$java.home/lib/jaxp.properties is read only once by the JSR-000206 Java™ API for XML Processing (“Specification”) imple-



mentation and its values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in `$java.home/lib/jaxp.properties` after it has been read for the first time.

If the property `javax.xml.validation.SchemaFactory:schemaLanguage` is present (where `schemaLanguage` is the parameter to `newInstance`), then its value is read as a `Class` name. The method will try to create a new instance of this `Class` by using the `ClassLoader`, and returns it if it is successfully created.

- The `ClassLoader` is asked for service provider provider-configuration files matching `javax.xml.validation.SchemaFactory` in the resource directory `META-INF/services`. See the JAR File Specification for file format and parsing rules. Each potential service provider is required to implement the method:

```
isSchemaLanguageSupported(String schemaLanguage)
```

The first service provider found in `ClassLoader` order that supports the specified schema language is returned.

- A platform default `SchemaFactory` is located in an implementation specific way. There must be a platform default `SchemaFactory` for W3C®, (World Wide Web Consortium) XML Schema.

If all lookups fail, then an `IllegalArgumentException` will be thrown.

### Tip

See `Properties.load(java.io.InputStream)` for exactly how a property file is parsed. In particular, colons ':' need to be escaped in a property file, so make sure schema language URIs are properly escaped. For example:

```
http\://www.w3.org/2001/XMLSchema=org.acme.XSSchemaFactory
```

## Streaming API for XML Plugability

JAXP defines a plugability mechanism to dynamically load compliant implementations of SAX and DOM parsers using the `javax.xml.parsers` and `javax.xml.transform` APIs. In an analogous manner, the StAX APIs define plugability mechanisms which allow applications to dynamically load compliant implementations of StAX.

See the discussion of `javax.xml.stream.XMLInputFactory`, `javax.xml.stream.XMLOutputFactory`, and `javax.xml.stream.XMLEventFactory` in the StAX Specification for further details.

## Examples

The following example demonstrates how a streaming reader can be used to transform parts of a document. Imagine, for the purposes of this somewhat contrived example, that you want to transform only those parts of your document that are labelled with the editorial status “draft”.

Begin by constructing a `Transformer`:

```
XMLInputFactory inputFactory = XMLInputFactory.newInstance();
```

```

FileInputStream styleis = new FileInputStream("inStyle.xml");
XMLStreamReader stylesheet = inputFactory.createXMLStreamReader(styleis);

Source style = new StAXSource(stylesheet);
TransformerFactory tf = TransformerFactory.newInstance();
Transformer t = tf.newTransformer(style);

```

Then create a reader for your input file:

```

FileInputStream fis = new FileInputStream("inFile.xml");
XMLStreamReader reader = inputFactory.createXMLStreamReader(fis);

```

Finally, walk through the input document using the pull API. Each time you find an element marked “draft”, transform it:

```

while(reader.hasNext()) {
    if (reader.getEventType() == XMLStreamReader.START_ELEMENT) {
        if ("draft".equals(reader.getAttributeValue(null, "status"))) {
            // Transform this...
            Source source = new StAXSource(reader);
            DOMResult result = new DOMResult();
            t.transform(source, result);
            // Do something with the result...
        } else {
            // This element doesn't need to be transformed
        }
    }
    reader.next();
}

```

Of course, it's also possible to simply transform entire documents using the event interfaces.

## Datatype Plugability

The Datatype Plugability classes allow an application programmer to obtain a variety of datatype objects. In order to obtain a datatype, a programmer first obtains an instance of the `DatatypeFactory`. The `DatatypeFactory` instance is obtained via one of the static `newInstance` methods of the `DatatypeFactory` class.

This method uses the following ordered lookup procedure to determine the `DatatypeFactory` implementation Class to load:

1. If the system property specified by `DATATYPEFACTORY_PROPERTY`, "javax.xml.datatype.DatatypeFactory", exists, a class with the name of the property's value is instantiated. Any Exception thrown during the instantiation process is wrapped as a `DatatypeConfigurationException`.
2. If the file `${JAVA_HOME}/lib/jaxp.properties` exists, it is loaded in a `Properties` Object. The `Properties` Object is then queried for the property as documented in the prior step and processed as documented in the prior step.

3. The services resolution mechanism is used, e.g. `META-INF/services/java.xml.datatype.DatatypeFactory`. Any `Exception` thrown during the instantiation process is wrapped as a `DatatypeConfigurationException`.
4. The final mechanism is to attempt to instantiate the `Class` specified by `DATATYPEFACTORY_IMPLEMENTATION_CLASS`. Any `Exception` thrown during the instantiation process is wrapped as a `DatatypeConfigurationException`.

The `DATATYPEFACTORY_IMPLEMENTATION_CLASS` constant is intended for JAXP implementors to specify a “class of last resort”. Application programmers should never refer to the constant in their programs. Instead, they should rely on the lookup procedure to locate an appropriate class.

## Thread Safety

Implementations of `SAXParser`, `DocumentBuilder`, `Transformer`, `Validator`, `ValidatorHandler`, `XPath`, and `XPathExpression` are not expected to be thread safe by this specification. This means that application programmers should not expect to be able to use an instance of any of them in more than one thread at a time without side effects. If a programmer is creating a multi-threaded application, they should make sure that only one thread has access to any instance at any given time.

Configuration of a `SAXParserFactory`, `DocumentBuilderFactory`, `TransformerFactory`, `SchemaFactory`, or `XPathFactory` is also not expected to be thread safe. This means that an application programmer should not allow an instance of any of these factories to have its setter methods accessed from more than one thread.

It is expected that the `newSAXParser` method of a `SAXParserFactory` implementation, the `newDocumentBuilder` method of a `DocumentBuilderFactory` and the `newTransformer` method of a `TransformerFactory` will be thread safe without side effects. This means that an application programmer should expect to be able to create transformer instances in multiple threads at once from a shared factory without side effects or problems. Note however that this guarantee can not automatically be extended to all the classes that a transformation might access. Consider, for example, the `URIResolver` or `ErrorHandler` classes. If an application is going to share instances of these classes across several transformation threads, the application must assure that they are thread-safe.

The `Schema` class is thread safe.

## Properties For Enabling Schema Validation

```
javax.xml.parsers.SAXParserFactory
```

The validating property must have the value `true` for any of the property strings defined below to take effect. Otherwise, the values of the properties defined below will be ignored. This value can be set by invoking

```
setValidating(true)
```

```
javax.xml.parsers.SAXParser
```

The `setProperty` method in `SAXParser` must support the property strings defined below to indicate the schema language and the source of the schema file(s) to the parser:

```
http://java.sun.com/xml/jaxp/properties/schemaLanguage
```

This property defines the schema language to be used for validation. The value of this property must be the URI of the schema language specification. To be compliant with this version of the specification, the

implementation must support the W3C XML Schema [<http://www.w3.org/2001/XMLSchema>] specification.

When `setValidating` is set to true and a schema language is set, then the parser must validate against that schema language only. For example if an application sets the `schemaLanguage` property to XML Schemas then the parser must try to validate against the XML schema only, even if the document has a DOCTYPE declaration that refers to a DTD.

```
http://java.sun.com/xml/jaxp/properties/schemaSource
```

The XML Schema Recommendation explicitly states that the inclusion of `schemaLocation` / `noNamespaceSchemaLocation` attributes in an instance document is only a hint; it does not mandate that these attributes must be used to locate schemas.

The `schemaSource` property lets the user set the schema(s) to validate against. If the target namespace of a schema specified using this property matches the target namespace of a schema occurring in `schemaLocation` attribute, the schema specified by the user using this property will be used and the instance document's `schemaLocation` attribute will be effectively ignored. However if the target namespace of any schema specified using this property doesn't match the target namespace of a schema occurring in the instance document, then the hint specified in the instance document will be used for validation. The acceptable value for this property must be one of the following:

- URI of the schema as a String
- `InputStream` with the contents of the schema
- `SAX InputSource`
- `File`
- an array of `Objects` with the contents being one of the types defined above. An array of `Objects` can be used only when the schema language has the ability to assemble a schema at runtime. When an array of `Objects` is passed it is illegal to have two schemas that share the same namespace.

If no target namespace is defined, then only one schema can be referenced by the property and it must work exactly the way `xsi:noNamespaceSchemaLocation` does.

It is illegal to set the `schemaSource` property if the `schemaLanguage` property has not been set. In that case, the implementation must throw a `SAXNotSupportedException` with a detailed message.

If the `schemaSource` property is set using a String, the parser must pass the value of the property to the `org.xml.sax.EntityResolver` with the `publicId` set to null.

```
javax.xml.parsers.DocumentBuilderFactory
```

The same property strings as described above for the `SAXParser` must be supported by `DocumentBuilderFactory.setAttribute` method.

When `setValidating` is set to true and a schema language is set then the parser must validate against that schema language only. For example if an application sets the schema language property to XML Schemas the parser must try to validate against the XML schema only, even if the document has a DOCTYPE declaration that refers to a DTD.

It is illegal to set the `schemaSource` property if the `schemaLanguage` property has not been set. In that case, the implementation must throw an `IllegalArgumentException` with a detailed message.

Note: None of the properties will take effect till the `setValidating(true)` has been called on the `SAXParserFactory` or the `DocumentBuilderFactory` that was used to create the `SAXParser` or the `DocumentBuilder`.

The table below shows the results of various configuration scenarios. In all cases, we assume that `setValidating(true)` has been called.

Document has DOC-TYPE?	schema language property set to XML Schemas?	schema source property set?	schema location attribute present in document?	Document Validated against	Schema file used
no	no	no	no	error as per JAXP 1.1 spec. Must have a DOC-TYPE declaration when validation is turned on.	N/A
no	no	no	yes	Error. Schema language must be set.	N/A
no	no	yes	yes / no	Error schema language must be set.	N/A
yes / no	yes	no	yes	xml schemas	schema files referred to using the schema location attributes present in the instance document
yes / no	yes	yes	no	xml schemas	schema file referred to in the <code>schemaSource</code> property
yes / no	yes	yes	yes	xml schemas	schema file referred to in the schema source property, if the target namespace matches. The schema file referred to in the schema location attribute is ignored only if the target namespace matches
yes	no	no	yes / no	DTD	DTD referred to in the DOC-TYPE
yes	no	yes	yes / no	Error. Schema source cannot be set without setting the schema language.	N/A

## Samples Using the Properties

Sax parser sample:

```
try {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    spf.setNamespaceAware(true);
    spf.setValidating(true);
    SAXParser sp = spf.newSAXParser();
    sp.setProperty("http://java.sun.com/xml/jaxp/properties/schemaLanguage",
        "http://www.w3.org/2001/XMLSchema");
}
```

```

    sp.setProperty("http://java.sun.com/xml/jaxp/properties/schemaSource",
        "http://www.example.com/Report.xsd");
    DefaultHandler dh = new DefaultHandler();
    sp.parse("http://www.example.com/foo.xml", dh);

} catch(Exception e) {
    e.printStackTrace();
}

```

DOM parser sample:

```

try {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setNamespaceAware(true);
    dbf.setValidating(true);
    dbf.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaLanguage",
        "http://www.w3.org/2001/XMLSchema");
    dbf.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaSource",
        "http://www.example.com/Report.xsd");
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document doc = db.parse("http://www.example.com/foo.xml");
} catch(Exception e) {
    e.printStackTrace();
}

```

## Recommended Implementation of Properties

It is recommended that parser implementations recognize properties defined in the form of a URI, as above. Such implementations avoid conflicts in the use of the feature and property strings among parser implementations. That is also the way in which SAX defines feature and property strings.

---

# Chapter 5. Conformance Requirements

This section describes the conformance requirements for implementations of this specification. Implementations that are accessed via the APIs defined here must implement these constraints, without exception, to provide a predictable environment for application development and deployment.

Note that applications may provide non-conformant implementations that are able to support the plugability mechanism defined in the specification, however the system default processor must meet the conformance requirements defined below.

## **All Implementations of JSR-000206 Java™ API for XML Processing (“Specification”) Must Support the Following Specifications:**

- Extensible Markup Language (XML) 1.1 [<http://www.w3.org/TR/xml11/>], Extensible Markup Language (XML) 1.0 (Third Edition) [<http://www.w3.org/TR/REC-xml/>], Extensible Markup Language (XML) 1.0 (Third Edition) Errata [<http://www.w3.org/XML/xml-V10-3e-errata>]
- Namespaces in XML 1.1 [<http://www.w3.org/TR/xml-names11/>], Namespaces in XML 1.0 [[spec.namespaces-1.0.link;](http://www.w3.org/TR/REC-xml-names/)], Namespaces in XML 1.0 Errata [[spec.namespaces-1.0-errata.link;](http://www.w3.org/TR/REC-xml-names/#Errata1)]
- XML Schema Part 1: Structures Second Edition [<http://www.w3.org/TR/xmlschema-1/>], XML Schema Part 1: Structures Second Edition Errata [<http://www.w3.org/2004/03/xmlschema-errata#Errata1>], XML Schema Part 2: Datatypes Second Edition [<http://www.w3.org/TR/xmlschema-2/>], XML Schema Part 2: Datatypes Second Edition Errata [<http://www.w3.org/2004/03/xmlschema-errata#Errata2>]
- XSL Transformations (XSLT) Version 1.0 [<http://www.w3.org/TR/xslt/>], XSL Transformations (XSLT) Version 1.0 Errata [<http://www.w3.org/1999/11/REC-xslt-19991116-errata>]
- XML Path Language (XPath) Version 1.0 [<http://www.w3.org/TR/xpath/>], XML Path Language (XPath) Version 1.0 Errata [<http://www.w3.org/1999/11/REC-xpath-19991116-errata>]
- XML Inclusions (XInclude) Version 1.0 [<http://www.w3.org/TR/xinclude/>]
- Document Object Model (DOM) Level 2 Core [<http://www.w3.org/TR/DOM-Level-2-Core/>], Document Object Model (DOM) Level 2 Traversal and Range, Document Object Model (DOM) Level 2 Events
- Document Object Model (DOM) Level 3 Core [<http://www.w3.org/TR/DOM-Level-3-Core/>], Document Object Model (DOM) Level 3 Load and Save
- Simple API for XML (SAX) 2.0.2 (sax2r3) [<http://sax.sourceforge.net/>], <http://sax.sourceforge.net/?selected=ext>

---

# Chapter 6. XML Inclusions (XInclude)

## What is XML Inclusions (XInclude)

JSR-000206 Java™ API for XML Processing (“Specification”) supports XInclude as defined in XML Inclusions (XInclude) Version 1.0 [<http://www.w3.org/TR/xinclude/>].

XInclude specifies a processing model and syntax for general purpose inclusion. Inclusion is accomplished by merging a number of XML information sets into a single composite Infoset. Specification of the XML documents (infosets) to be merged and control over the merging process is expressed in XML-friendly syntax (elements, attributes, URI references).

— XML Inclusions (XInclude) Version 1.0, Abstract  
[<http://www.w3.org/TR/xinclude/#abstract>]

## Implementation Required by JSR-000206 Java™ API for XML Processing (“Specification”)

**JSR-000206 Java™ API for XML Processing (“Specification”) implements *XML Inclusions (XInclude) Version 1.0* with the following limitations**

- at the time this specification was developed, there was no standard fragment identifier syntax for “application/xml” resources
- supports XPointers using the *XPointer Framework* and *XPointer element() scheme*
- no other XPointer schemes or addressing mechanisms are supported and it is an error to use an other XPointer scheme or addressing mechanism, the error is signaled by throwing a `java.lang.Exception` for DOM processing and an `org.xml.sax.SAXParseException` for SAX processing

XInclude processing defaults to `false`. See Javadoc for `javax.xml.parsers.DocumentBuilderFactory` and `javax.xml.parsers.SAXParserFactory` for methods to enable/disable/query the state of XInclude processing.



---

# Chapter 7. JSR-000206 Java™ API for XML Processing (“Specification”)

## Specification and Implementation

### Version Information

## JSR-000206 Java™ API for XML Processing (“Specification”) Specification Version Information

**JSR-000206 Java™ API for XML Processing (“Specification”) specification version information is made available via `java.lang.Package` methods**

- `public String getSpecificationTitle();`

Return the title of the specification that this package implements. `null` is returned if it is not known.

- `public String getSpecificationVersion();`

Returns the version number of the specification that this package implements. This version string must be a sequence of positive decimal integers separated by "."s and may have leading zeros. When version strings are compared the most significant numbers are compared. `null` is returned if it is not known.

- `public String getSpecificationVendor();`

Returns the name of the organization, vendor, or company that owns and maintains the specification of the classes that implement this package. `null` is returned if it is not known.

This version information is retrieved and made available by the `ClassLoader` instance that loaded the class(es). Typically, it is stored in the manifest that is distributed with the classes. Implementations are required to provide this information with the following values:

**Table 7.1. JSR-000206 Java™ API for XML Processing (“Specification”) Specification Version Information**

Specification Title	JSR-000206 Java™ API for XML Processing (“Specification”)
Specification Vendor	Sun Microsystems, Inc.
Specification Version	1.4

Sample `META-INF/MANIFEST.MF` entries to provide this information would be:

Specification-Title : JSR-000206 Java™ API for XML Processing (“Specification”)  
Specification-Vendor : Sun Microsystems, Inc.  
Specification-Version : 1.4

# JSR-000206 Java™ API for XML Processing (“Specification”) Implementation Version Information

**JSR-000206 Java™ API for XML Processing (“Specification”) implementation version information is made available via `java.lang.Package` methods**

- `public String getImplementationTitle();`

Return the title of this package. `null` is returned if it is not known.

- `public String getImplementationVersion();`

Returns the version of this implementation. It consists of any string assigned by the vendor of this implementation and does not have any particular syntax specified or expected by the Java runtime. It may be compared for equality with other package version strings used for this implementation by this vendor for this package. `null` is returned if it is not known.

- `public String getImplementationVendor();`

Returns the name of the organization, vendor or company that provided this implementation.

This version information is retrieved and made available by the `ClassLoader` instance that loaded the class(es). Typically, it is stored in the manifest that is distributed with the classes. Implementations are required to provide this information with reasonable values:

Implementation Title  
Implementation Vendor  
Implementation Version

Sample `META-INF/MANIFEST.MF` entries to provide this information would be:

Implementation-Title : My Implementation Title  
Implementation-Vendor : My Implementation Vendor  
Implementation-Version : My Implementation Version

See the *JAR File Specification*, `${JAVA_HOME}/docs/guide/jar/jar.html`, for detailed format information.

---

# Chapter 8. Constants

**Table 8.1. Constants in `javax.xml.datatype.DatatypeFactory`**

Type	Name	Value
<code>java.lang.String</code>	<code>DATATYPEFACTORY_PROPERTY</code>	<code>"javax.xml.datatype.DatatypeFactory"</code>
<code>java.lang.String</code>	<code>DATATYPEFACTORY_IMPLEMENTATION_CLASS</code>	<i>Implementation defined<sup>a</sup></i>

<sup>a</sup>Implementers should specify the name of an appropriate class to be instantiated if no other implementation resolution mechanism succeeds. Users should not refer to this field; it is intended only to document a factory implementation detail.

**Table 8.2. Constants in javax.xml.datatype.DatatypeConstants**

Type	Name	Value
int	JANUARY	1
int	FEBRUARY	2
int	MARCH	3
int	APRIL	4
int	MAY	5
int	JUNE	6
int	JULY	7
int	AUGUST	8
int	SEPTEMBER	9
int	OCTOBER	10
int	NOVEMBER	11
int	DECEMBER	12
int	LESSER	-1
int	EQUAL	0
int	GREATER	1
int	INDETERMINATE	2
int	FIELD_UNDEFINED	Integer.MIN_VALUE
javax.xml.datatype.DatatypeConstants.FIELD	YEARS	new Field("YEARS", 0)
javax.xml.datatype.DatatypeConstants.FIELD	MONTHS	new Field("MONTHS", 1)
javax.xml.datatype.DatatypeConstants.FIELD	DAYS	new Field("DAYS", 2)
javax.xml.datatype.DatatypeConstants.FIELD	HOURS	new Field("HOURS", 3)
javax.xml.datatype.DatatypeConstants.FIELD	MINUTES	new Field("MINUTES", 4)
javax.xml.datatype.DatatypeConstants.FIELD	SECONDS	new Field("SECONDS", 5)
javax.xml.namespace.QName	DATETIME	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "dateTime")
javax.xml.namespace.QName	TIME	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "time")
javax.xml.namespace.QName	DATE	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "date")
javax.xml.namespace.QName	GYEARMONTH	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gYearMonth")

Type	Name	Value
javax.xml.namespace.QName	GMONTHDAY	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gMonthDay")
javax.xml.namespace.QName	GYEAR	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gYear")
javax.xml.namespace.QName	GMONTH	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gMonth")
javax.xml.namespace.QName	GDAY	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "gDay")
javax.xml.namespace.QName	DURATION	new QName(XMLConstants.W3C_XML_SCHEMA_NS_URI, "duration")
javax.xml.namespace.QName	DURATION_DAYTIME	new QName(XMLConstants.W3C_XPATH_DATA_TYPE_NS_URI, "dayTimeDuration")
javax.xml.namespace.QName	DURATION_YEAR-MONTH	new QName(XMLConstants.W3C_XPATH_DATA_TYPE_NS_URI, "yearMonthDuration")
int	MAX_TIMEZONE_OFFSET	-14 * 60
int	MIN_TIMEZONE_OFFSET	14 * 60

**Table 8.3. Constants in javax.xml.transform.dom.DOMResult**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.dom.DOMResult/feature"

**Table 8.4. Constants in javax.xml.transform.dom.DOMSource**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.dom.DOMSource/feature"

**Table 8.5. Constants in javax.xml.transform.sax.SAXResult**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.sax.SAXResult/feature"

**Table 8.6. Constants in javax.xml.transform.sax.SAXSource**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.sax.SAXSource/feature"

**Table 8.7. Constants in javax.xml.transform.sax.SAXTransformerFactory**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.sax.SAXTransformerFactory/feature"
java.lang.String	FEATURE_XMLFILTER	"http://javax.xml.transform.sax.SAXTransformerFactory/feature/xmlfilter"

**Table 8.8. Constants in javax.xml.transform.stream.StreamResult**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.stream.StreamResult/feature"

**Table 8.9. Constants in javax.xml.transform.stream.StreamSource**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.stream.StreamSource/feature"

**Table 8.10. Constants in javax.xml.transform.OutputKeys**

Type	Name	Value
java.lang.String	METHOD	"method"
java.lang.String	VERSION	"version"
java.lang.String	ENCODING	"encoding"
java.lang.String	OMIT_XML_DECLARATION	"omit-xml-declaration"
java.lang.String	STANDALONE	"standalone"
java.lang.String	DOCTYPE_PUBLIC	"doctype-public"
java.lang.String	DOCTYPE_SYSTEM	"doctype-system"
java.lang.String	CDATA_SECTION_ELEMENTS	"cdata-section-elements"
java.lang.String	INDENT	"indent"
java.lang.String	MEDIA_TYPE	"media-type"

**Table 8.11. Constants in javax.xml.transform.Result**

Type	Name	Value
java.lang.String	PI_DISABLE_OUTPUT_ESCAPING	"javax.xml.transform.disable-output-escaping"
java.lang.String	PI_ENABLE_OUTPUT_ESCAPING	"javax.xml.transform.enable-output-escaping"

**Table 8.12. Constants in javax.xml.transform.stax.StAXResult**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.stax.StAXResult/feature"

**Table 8.13. Constants in javax.xml.transform.stax.StAXSource**

Type	Name	Value
java.lang.String	FEATURE	"http://javax.xml.transform.stax.StAXSource/feature"

**Table 8.14. Constants in javax.xml.xpath.XPathConstants**

Type	Name	Value
javax.xml.namespace.QName	NUMBER	new QName("http://www.w3.org/1999/XSL/Transform", "NUMBER")
javax.xml.namespace.QName	STRING	new QName("http://www.w3.org/1999/XSL/Transform", "STRING")
javax.xml.namespace.QName	BOOLEAN	new QName("http://www.w3.org/1999/XSL/Transform", "BOOLEAN")
javax.xml.namespace.QName	NODESET	new QName("http://www.w3.org/1999/XSL/Transform", "NODESET")
javax.xml.namespace.QName	NODE	new QName("http://www.w3.org/1999/XSL/Transform", "NODE")
java.lang.String	DOM_OBJECT_MODEL	"http://java.sun.com/jaxp/xpath/dom"

**Table 8.15. Constants in javax.xml.xpath.XPathFactory**

Type	Name	Value
java.lang.String	DEFAULT_PROPERTY_NAME	"javax.xml.xpath.XPathFactory"
java.lang.String	DEFAULT_OBJECT_MODEL_URI	"http://java.sun.com/jaxp/xpath/dom"

**Table 8.16. Constants in javax.xml.stream.XMLInputFactory**

Type	Name	Value
java.lang.String	IS_NAMESPACE_AWARE	"javax.xml.stream.isNamespaceAware"
java.lang.String	IS_VALIDATING	"javax.xml.stream.isValidating"
java.lang.String	IS_COALESCING	"javax.xml.stream.isCoalescing"
java.lang.String	IS_REPLACING_ENTITY_REFERENCES	"javax.xml.stream.isReplacingEntityReferences"
java.lang.String	IS_SUPPORTING_EXTERNAL_ENTITIES	"javax.xml.stream.isSupportingExternalEntities"
java.lang.String	SUPPORT_DTD	"javax.xml.stream.supportDTD"
java.lang.String	REPORTER	"javax.xml.stream.reporter"
java.lang.String	RESOLVER	"javax.xml.stream.resolver"
java.lang.String	ALLOCATOR	"javax.xml.stream allocator"

**Table 8.17. Constants in javax.xml.stream.XMLOutputFactory**

Type	Name	Value
java.lang.String	IS_REPAIRING_NAMESPACES	"javax.xml.stream.isRepairingNamespaces"

**Table 8.18. Constants in javax.xml.stream.XMLStreamConstants**

Type	Name	Value
int	START_ELEMENT	1
int	END_ELEMENT	2
int	PROCESSING_INSTRUCTION	3
int	CHARACTERS	4
int	COMMENT	5
int	SPACE	6
int	START_DOCUMENT	7
int	END_DOCUMENT	8
int	ENTITY_REFERENCE	9
int	ATTRIBUTE	10
int	DTD	11
int	CDATA	12
int	NAMESPACE	13
int	NOTATION_DECLARATION	14
int	ENTITY_DECLARATION	15

**Table 8.19. Constants in javax.xml.XMLConstants**

Type	Name	Value
java.lang.String	NULL_NS_URI	""
java.lang.String	DEFAULT_NS_PREFIX	""
java.lang.String	XML_NS_URI	"http://www.w3.org/XML/1998/namespace"
java.lang.String	XML_NS_PREFIX	"xml"
java.lang.String	XMLNS_ATTRIBUTE_NS_URI	"http://www.w3.org/2000/xmlns/"
java.lang.String	XMLNS_ATTRIBUTE	"xmlns"
java.lang.String	W3C_XML_SCHEMA_NS_URI	"http://www.w3.org/2001/XMLSchema"
java.lang.String	W3C_XML_SCHEMA_INSTANCE_NS_URI	"http://www.w3.org/2001/XMLSchema-instance"
java.lang.String	W3C_XPATH_DATATYPE_NS_URI	"http://www.w3.org/2003/11/xpath-datatypes"
java.lang.String	XML_DTD_NS_URI	"http://www.w3.org/TR/REC-xml"
java.lang.String	RELAXNG_NS_URI	"http://relaxng.org/ns/structure/1.0"
java.lang.String	FEATURE_SECURE_PROCESSING	"http://javax.xml.XMLConstants/feature/secure-processing"



---

# Chapter 9. Changes Since JAXP 1.3

The following sections summarize the significant changes introduced in JAXP 1.4. For additional details, please see the appropriate JavaDoc.

## Package `javax.xml.datatype`

### Class `DatatypeFactory`

- Changed `DATATYPEFACTORY_IMPLEMENTATION_CLASS` to `com.sun.org.apache.xerces.internal.jaxp.datatype.DatatypeFactoryImpl`.
- Added `newInstance(String, ClassLoader)` method.

### Class `Duration`

- Return `false` if the argument to `equals()` is `null` instead of throwing `NullPointerException`.

### Class `XMLGregorianCalendar`

- Return `false` if the argument to `equals()` is `null` instead of throwing `NullPointerException`.

## Package `javax.xml.parsers`

### Class `DocumentBuilderFactory`

- Added `newInstance(String, ClassLoader)` method.

### Class `SAXParserFactory`

- Added `newInstance(String, ClassLoader)` method.

## Package `javax.xml.stream`

Added the `javax.xml.stream` package to support StAX. The package consists of the following interfaces:

- `EventFilter`
- `Location`
- `StreamFilter`
- `XMLEventReader`
- `XMLEventWriter`
- `XMLReporter`
- `XMLResolver`

- XMLStreamConstants
- XMLStreamReader
- XMLStreamWriter

Classes:

- XMLEventFactory
- XMLInputFactory
- XMLOutputFactory

Exceptions:

- XMLStreamException

And errors:

- FactoryConfigurationError

## Package `javax.xml.stream.events`

Added the `javax.xml.stream.events` package to support StAX. The package consists of the following interfaces:

- Attribute
- Characters
- Comment
- DTD
- EndDocument
- EndElement
- EntityDeclaration
- EntityReference
- Namespace
- NotationDeclaration
- ProcessingInstruction
- StartDocument
- StartElement
- XMLEvent

## Package `javax.xml.stream.util`

Added the `javax.xml.stream.util` package to support StAX. The package consists of the following interfaces:

- `XMLEventAllocator`
- `XMLEventConsumer`

And classes:

- `EventReaderDelegate`
- `StreamReaderDelegate`

## Package `javax.xml.transform`

### Class `ErrorListener`

- Clarified the semantics of `fatalError()`.

### Class `Transformer`

- Clarified the semantics of `getOutputProperty()` with respect to properties that have not been set explicitly with either `setOutputProperty()` or `xsl:output` in the stylesheet.

### Class `TransformerFactory`

- Added `newInstance(String, ClassLoader)` method.

## Package `javax.xml.transform.stax`

Added the `javax.xml.transform.stax` package to support StAX. The package consists of the following classes:

- `StAXResult`
- `StAXSource`

## Package `javax.xml.validation`

### Class `Schema`

- Clarified that the features set on the `SchemaFactory` should be passed to the `Validator` created with `newValidator()`.
- Clarified that the features set on the `SchemaFactory` should be passed to the `ValidatorHandler` created with `newValidatorHandler()`.

## Class SchemaFactory

- Added `newInstance(String, ClassLoader)` method.
- Clarified that the features set on the `SchemaFactory` should be passed to the Schemas created with `newSchema()`. Included a note to implementors and developers about the subtleties associated with `newSchema()` in this context.
- Clarified that the inputs to `newSchema(Source[])` are expected to be XML documents or elements.
- Updated documentation of `setFeature()` to clarify that the features set on the `SchemaFactory` should be passed to the Schemas created with this factory and by extension to the `Validator` and `ValidatorHandlers` created from that Schema.

## Class SchemaFactoryLoader

The `SchemaFactoryLoader` class was created during the JAXP 1.3 development process. Shortly before JAXP 1.3 was finished, the factory mechanisms associated with validation were changed, and this class was removed. Unfortunately, the file was left in a repository and it slipped into the *Java 2 Platform Standard Edition 5.0 API Specification*.

Since the class can neither be added to JAXP 1.3 nor removed from the Java 5.0 API Specification due to backwards compatibility issues, it is being added to JAXP 1.4. The class is harmless and should not be used. It is being added simply to avoid the confusion that arises when developers notice that it's defined in the platform but not in JAXP.

*Do not use this class.*

## Class TypeInfoProvider

- Extended the semantics of `getElementTypeInfo()`, allowing it to be called from either the `startElement` event *or* the `endElement` event. This allows the API to support W3C XML Schema union types more usefully.

When W3C XML Schema validation is being performed, in the case where an element has a union type, the `TypeInfo` returned by a call to `getElementTypeInfo()` from the `startElement` event will be the union type. The `TypeInfo` returned by a call from the `endElement` event will be the actual member type used to validate the element.

## Class Validator

- The `Validator` may now accept `Sources` other than `DOMSource` and `SAXSource`; for example, `StAXSource`.
- Clarified that the inputs to `validate()` are expected to be XML documents or elements.

## Package javax.xml.xpath

### Class XPath

- Clarified that it is the `XPathVariableResolver` in effect at compile time that is used to resolve any variables that appear in the expression.

## Class XPathFactory

- Added `newInstance(String, ClassLoader)` method.

---

# Chapter 10. Colophon

## Talk the Talk, Walk the Walk

JSR-000206 Java™ API for XML Processing (“Specification”) was authored in XML using the DocBook [<http://docbook.org/>] DTD.

The XML was transformed into XHTML, HTML and XSL Formatting Objects using the DocBook style sheets. The XSL Formatting Objects were then transformed into PDF.