

# Oracle9i

XML Reference

Release 1 (9.0.1)

June 2001

Part No. A88899-01

ORACLE®

---

Oracle9i XML Reference, Release 1 (9.0.1)

Part No. A88899-01

Copyright © 2001, Oracle Corporation. All rights reserved.

Primary Author: Chitra Sharma

Contributing Author: Jack Melnick

Contributors: Muralidhar Krishnaprasad, Anjana Manian, Visar Nimani, Mark Scardina, Ian Macky, Vikram Yavagal

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

**Restricted Rights Notice** Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle, Oracle Call Interface, Oracle Forms, and SQL\*Plus are registered trademarks of Oracle Corporation. Net8, Oracle7, Oracle7 Server, Oracle8, Oracle8 Server, Oracle8i, Oracle9i, PL/SQL, Pro\*C, Pro\*C/C++, Pro\*Cobol, and SQL\*Net are trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

---

---

# Contents

<b>Send Us Your Comments .....</b>	<b>xiii</b>
<b>Preface.....</b>	<b>xv</b>
Audience .....	xvi
Organization.....	xvi
Related Documentation .....	xvi
Conventions.....	xvii
Documentation Accessibility .....	xix
.....	xx
<b>What's New in XML Reference? .....</b>	<b>xxi</b>
Oracle9i Release 1 (9.0.1) New Features in XML Reference .....	xxii
<b>Part I  XDK for Java Packages</b>	
<b>1  Package oracle.xml.parser.v2</b>	
<b>Description.....</b>	<b>1-2</b>
<b>AttrDecl.....</b>	<b>1-4</b>
<b>DefaultXMLDocumentHandler.....</b>	<b>1-9</b>
<b>DOMParser .....</b>	<b>1-15</b>
<b>DTD .....</b>	<b>1-22</b>
<b>ElementDecl.....</b>	<b>1-30</b>
<b>NodeFactory.....</b>	<b>1-36</b>
<b>NSName.....</b>	<b>1-40</b>

<b>NSResolver</b> .....	1-42
<b>oraxsl</b> .....	1-43
<b>SAXAttrList</b> .....	1-45
<b>SAXParser</b> .....	1-51
<b>XMLAttr</b> .....	1-55
<b>XMLCDATA</b> .....	1-63
<b>XMLComment</b> .....	1-66
<b>XMLDocument</b> .....	1-69
<b>XMLDocumentFragment</b> .....	1-83
<b>XMLDocumentHandler</b> .....	1-86
<b>XMLElement</b> .....	1-91
<b>XMLEntityReference</b> .....	1-103
<b>XMLNode</b> .....	1-105
<b>XMLParseException</b> .....	1-118
<b>XMLParser</b> .....	1-122
<b>XMLPI</b> .....	1-128
<b>XMLText</b> .....	1-131
<b>XMLToken</b> .....	1-135
<b>XMLTokenizer</b> .....	1-141
<b>XSLException</b> .....	1-147
<b>XSLProcessor</b> .....	1-148
<b>XSLStylesheet</b> .....	1-154

## 2 Package oracle.xml.classgen

<b>CGDocument</b> .....	2-2
<b>CGNode</b> .....	2-4
<b>CGXSDElement</b> .....	2-9
<b>DTDClassGenerator</b> .....	2-12
<b>InvalidContentException</b> .....	2-14
<b>oracg</b> .....	2-15
<b>SchemaClassGenerator</b> .....	2-16

## 3 Package oracle.xml.xsql

<b>Description</b> .....	3-2
<b>Res</b> .....	3-3

<b>XSQLActionHandler</b> .....	3-13
<b>XSQLActionHandlerImpl</b> .....	3-15
<b>XSQLCommandLine</b> .....	3-17
<b>XSQLDiagnostic</b> .....	3-18
<b>XSQLHttpUtil</b> .....	3-20
<b>XSQLPageRequest</b> .....	3-22
<b>XSQLPageRequestImpl</b> .....	3-29
<b>XSQLParserHelper</b> .....	3-38
<b>XSQLRequest</b> .....	3-40
<b>XSQLServlet</b> .....	3-46
<b>XSQLServletPageRequest</b> .....	3-49
<b>XSQLStylesheetProcessor</b> .....	3-53
<b>XSQLUtil</b> .....	3-55

## 4 Transviewer Beans

<b>Package oracle.xml.async</b> .....	4-2
<b>DOMBuilder</b> .....	4-3
<b>DOMBuilderBeanInfo</b> .....	4-14
<b>DOMBuilderErrorEvent</b> .....	4-16
<b>DOMBuilderErrorListener</b> .....	4-18
<b>DOMBuilderEvent</b> .....	4-19
<b>DOMBuilderListener</b> .....	4-21
<b>ResourceManager</b> .....	4-23
<b>XSLTransformer</b> .....	4-25
<b>XSLTransformerBeanInfo</b> .....	4-30
<b>XSLTransformerErrorEvent</b> .....	4-32
<b>XSLTransformerErrorListener</b> .....	4-34
<b>XSLTransformerEvent</b> .....	4-35
<b>XSLTransformerListener</b> .....	4-37
<b>Package oracle.xml.dbviewer</b> .....	4-38
<b>DBViewer</b> .....	4-38
<b>Package oracle.xml.dbviewer</b> .....	4-53
<b>DBViewerBeanInfo</b> .....	4-53
<b>Package oracle.xml.srcviewer</b> .....	4-54
<b>XMLSourceView</b> .....	4-54

<b>Package oracle.xml.srcviewer</b> .....	4-66
<b>XMLSourceViewBeanInfo</b> .....	4-66
<b>Package oracle.xml.transviewer</b> .....	4-67
<b>DBAccess</b> .....	4-67
<b>DBAccessBeanInfo</b> .....	4-74
<b>XMLTransformPanel</b> .....	4-75
<b>XMLTransformPanelBeanInfo</b> .....	4-76
<b>XMLTransViewer</b> .....	4-77
<b>Package oracle.xml.treeviewer</b> .....	4-78
<b>XMLTreeView</b> .....	4-78
<b>XMLTreeViewBeanInfo</b> .....	4-81

## 5 Package oracle.XML.parser.schema

<b>XMLSchema</b> .....	5-2
<b>XSDBuilder</b> .....	5-3
<b>XSDException</b> .....	5-8

## Part II XDK for PL/SQL Packages

### 6 XML Parser for PL/SQL

<b>PL/SQL Parser APIs</b> .....	6-2
<b>Types and Functions</b> .....	6-3
<b>Parser Interface Type Description</b> .....	6-5
<b>Function Prototypes</b> .....	6-5
<b>Extensible Stylesheet Language (XSL) Package Processor APIs</b> .....	6-13
<b>Functions</b> .....	6-14
<b>W3C Document Object Model (DOM) APIs</b> .....	6-21
<b>Types</b> .....	6-23
<b>DOM Node Types</b> .....	6-23
<b>DOM Exception Types</b> .....	6-23
<b>DOM Interface Types</b> .....	6-24
<b>Methods</b> .....	6-25
<b>Node Methods</b> .....	6-25
<b>Named Node Map Methods</b> .....	6-31

<b>Node List Methods</b> .....	6-33
<b>Attr Methods</b> .....	6-34
<b>C Data Section Methods</b> .....	6-36
<b>Character Data Methods</b> .....	6-37
Comment Methods.....	6-38
DOM Implementation Methods.....	6-38
Document Fragment Methods.....	6-39
Document Type Methods.....	6-39
<b>Element Methods</b> .....	6-42
<b>Entity Methods</b> .....	6-46
<b>Entity Reference Methods</b> .....	6-47
<b>Notation Methods</b> .....	6-48
<b>Processing Instruction Methods</b> .....	6-49
<b>Text Methods</b> .....	6-49
<b>Document Methods</b> .....	6-51
<b>Method Prototypes</b> .....	6-56
<b>Node Methods</b> .....	6-56
<b>Named Node Map Methods</b> .....	6-65
<b>Node List Methods</b> .....	6-66
<b>Attr Methods</b> .....	6-67
<b>C Data Section Methods</b> .....	6-69
<b>Character Data Methods</b> .....	6-70
<b>Comment Methods</b> .....	6-71
<b>DOM Implementation Methods</b> .....	6-72
<b>Document Fragment Methods</b> .....	6-73
<b>Document Type Methods</b> .....	6-74
<b>Element Methods</b> .....	6-80
<b>Entity Methods</b> .....	6-84
<b>Entity Reference Methods</b> .....	6-85
<b>Notation Methods</b> .....	6-86
<b>Processing Instruction Methods</b> .....	6-87
<b>Text Methods</b> .....	6-88
<b>Document Methods</b> .....	6-89
<b>Interface org.w3c.dom.Attr</b> .....	6-98
<b>Interface org.w3c.dom.CDATASection</b> .....	6-100

Interface org.w3c.dom.CharacterData.....	6-101
Interface org.w3c.dom.Comment.....	6-105
Interface org.w3c.dom.Document.....	6-105
Interface org.w3c.dom.DocumentFragment.....	6-110
Interface org.w3c.dom.DocumentType.....	6-111
Class org.w3c.dom.DOMException.....	6-113
Interface org.w3c.dom.DOMImplementation.....	6-114
Interface org.w3c.dom.Element.....	6-115
Interface org.w3c.dom.Entity.....	6-120
Interface org.w3c.dom.EntityReference.....	6-122
Interface org.w3c.dom.NamedNodeMap.....	6-122
Interface org.w3c.dom.Node.....	6-125
Interface org.w3c.dom.NodeList.....	6-133
Interface org.w3c.dom.Notation.....	6-134
Interface org.w3c.dom.ProcessingInstruction.....	6-135
Interface org.w3c.dom.Text.....	6-136
.....	6-137

## Part III XDK for C Packages

### 7 XML Parser for C

Parser APIs.....	7-2
Calling Sequence.....	7-3
Memory.....	7-3
Thread Safety.....	7-3
Function/Method Index.....	7-3
Functions.....	7-5
XSLT API.....	7-12
Functions.....	7-14
Function Prototypes.....	7-15
W3C SAX APIs.....	7-15
Data Structures and Types.....	7-17
Non-SAX Callback Functions.....	7-18
Function Prototypes.....	7-19
W3C DOM APIs.....	7-24



<b>Function Prototypes</b> .....	7-29
<b>Namespace APIs</b> .....	7-74
<b>Data Structures and Types</b> .....	7-76
<b>Functions</b> .....	7-77
<b>Data Structure and Type Description</b> .....	7-78
<b>Function Prototypes</b> .....	7-79
<b>Datatypes</b> .....	7-83

## 8 XML Schema Processor for C

Schema APIs.....	8-2
Function/Method Index.....	8-2
<b>Functions</b> .....	8-2

## Part IV XDK for C++ Packages

### 9 XML Parser for C++

<b>Class: Attr</b> .....	9-2
<b>Class: CDATASection</b> .....	9-5
<b>Class: Comment</b> .....	9-6
<b>Class: Document</b> .....	9-7
<b>Class: DocumentType</b> .....	9-12
<b>Class: DOMImplementation</b> .....	9-14
<b>Class: Element</b> .....	9-15
<b>Class: Entity</b> .....	9-20
<b>Class: EntityReference</b> .....	9-22
<b>Class: NamedNodeMap</b> .....	9-23
<b>Class: Node</b> .....	9-26
<b>Class: NodeList</b> .....	9-38
<b>Class: Notation</b> .....	9-39
<b>Class: ProcessingInstruction</b> .....	9-41
<b>Class: Text</b> .....	9-43
<b>Class: XMLParser</b> .....	9-44
<b>C++ SAX API</b> .....	9-50
SAX callback structure .....	9-50

C++ DOM API's .....	9-56
<b>10 Oracle XML Class Generator (C++)</b>	
Overview of the XML Class Generator for C++.....	10-2
Input .....	10-2
Output .....	10-2
Relevant XML Standards.....	10-3
sample/Example usage .....	10-3
Class: XMLClassGenerator 10-4	
METHODS .....	10-4
generate .....	10-4
Class: generated 10-5	
METHOD INDEX .....	10-6
METHOD .....	10-7
Constructors .....	10-7
<b>11 XML Schema Processor for C++</b>	
Schema APIs .....	11-2
Function/Method Index .....	11-2
Functions.....	11-2
<b>Part V XML-SQL Utility (XSU) Packages</b>	
<b>12 Oracle XML SQL Utility (XSU) Java API</b>	
OracleXMLQuery .....	12-2
OracleXMLSave.....	12-18
OracleXMLSQLException .....	12-32
OracleXMLSQLNoRowsException .....	12-35
<b>13 XML SQL Utility (XSU) PL/SQL API</b>	
DBMS_XMLQuery .....	13-2
Types:.....	13-2
Constants:.....	13-2
Function and Procedure Index: .....	13-3

Functions and Procedures: .....	13-5
<b>DBMS_XMLSave</b> .....	13-13
Types: .....	13-13
Constants: .....	13-13
Function and Procedure Index: .....	13-13
Functions and Procedures: .....	13-15

## Part VI XML Support

### 14 XML Support

<b>DBMS_XMLGEN</b> .....	14-2
DBMS_XMLGEN Procedures and Functions.....	14-4
<b>DBMS_XMLGEN Type definitons</b> .....	14-6
<b>Functions Prototypes</b> .....	14-6
<b>URI Support</b> .....	14-13
UriType .....	14-13
<b>Member functions</b> .....	14-14
<b>Function prototypes</b> .....	14-15
HttpUritype .....	14-16
<b>Member functions</b> .....	14-16
<b>Function prototypes</b> .....	14-17
DbUritype .....	14-19
<b>Member functions</b> .....	14-20
<b>Function prototypes</b> .....	14-21
UriFactory Package .....	14-22
<b>Package functions</b> .....	14-24
<b>Function prototypes</b> .....	14-24
<b>XMLType</b> .....	14-28
<b>Member functions</b> .....	14-28
<b>Function prototypes</b> .....	14-29

## Index



---

---

# Send Us Your Comments

**Oracle9i XML Reference, Release 1 (9.0.1)**

**Part No. A88899-01**

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: [infodev\\_us@oracle.com](mailto:infodev_us@oracle.com)
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:

Oracle Corporation  
Server Technologies Documentation  
500 Oracle Parkway, Mailstop 4op11  
Redwood Shores, CA 94065  
USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.



---

# Preface

This preface discusses the following topics:

- [Audience](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)
- [Documentation Accessibility](#)

## Audience

Oracle9i XML Reference presents interfaces, classes, methods, packages and exceptions are summarized in Java, PL/SQL, C and C ++. XSU Java and PL/SQL APIs are referenced as well. The book also has a XML Support Section.

## Organization

See the table of contents for a listing of the various sections.

## Related Documentation

For more information, programmers should see:

- *Oracle9i JDBC Developer's Guide and Reference*
- *Oracle9i Application Developer's Guide - Fundamentals*
- *Oracle9i Application Developer's Guide - Advanced Queuing*
- *Oracle9i Data Cartridge Developer's Guide.*
- *Oracle9i Application Developer's Guide - XML*
- *Oracle9i Case Studies - XML Applications*

In North America, printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

<http://www.oraclebookshop.com/>

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

<http://technet.oracle.com/membership/index.htm>

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at



## Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- [Conventions in Text](#)
- [Conventions in Code Examples](#)

### Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
<b>Bold</b>	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an <b>index-organized table</b> .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> reside on the same disk.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values.  <b>Note:</b> Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter <code>sqlplus</code> to open SQL*Plus.  The password is specified in the <code>orapwd</code> file.  Back up the datafiles and control files in the <code>/disk1/oracle/dbs</code> directory.  The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table.  Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> .  Connect as <code>oe</code> user.  The <code>JRepUtil</code> class implements these methods.
<i>lowercase monospace (fixed-width font) italic</i>	Lowercase monospace italic font represents placeholders or variables.	You can specify the <i>parallel_clause</i> .  Run <code>Uold_release.SQL</code> where <i>old_release</i> refers to the release you installed prior to upgrading.

## Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL\*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	<code>DECIMAL (digits [ , precision ])</code>
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	<code>{ENABLE   DISABLE}</code>
	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	<code>{ENABLE   DISABLE}</code> <code>[COMPRESS   NOCOMPRESS]</code>

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> <li>That we have omitted parts of the code that are not directly related to the example</li> <li>That you can repeat a portion of the code</li> </ul>	<pre>CREATE TABLE ... AS subquery;  SELECT col1, col2, ... , coln FROM employees;</pre>
.	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2); acct      CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. <b>Note:</b> Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

## Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading

technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

---

---

# What's New in XML Reference?

The following sections describe the new features in Oracle9i:

- [Oracle9i Release 1 \(9.0.1\) New Features in XML Reference](#)

## Oracle9i Release 1 (9.0.1) New Features in XML Reference

### See Also:

- [Chapter 1, "Package oracle.xml.parser.v2"](#)
- [Chapter 2, "Package oracle.xml.classgen"](#)
- [Chapter 3, "Package oracle.xml.xsql"](#)
- [Chapter 4, "Transviewer Beans"](#)
- [Chapter 5, "Package oracle.XML.parser.schema"](#)
- [Chapter 12, "Oracle XML SQL Utility \(XSU\) Java API"](#)
- [Chapter 14, "XML Support"](#)

# Part I

---

## XDK for Java Packages

This section contains the following chapters:

- [Chapter 1, "Package oracle.xml.parser.v2"](#)
- [Chapter 2, "Package oracle.xml.classgen"](#)
- [Chapter 3, "Package oracle.xml.xsql"](#)
- [Chapter 4, "Transviewer Beans"](#)
- [Chapter 5, "Package oracle.XML.parser.schema"](#)





---

---

# Package oracle.xml.parser.v2

## Description

---

### Class Summary

---

#### Interfaces

<code>NSName</code>	This interface provides Namespace support for Element and Attr names
<code>NSResolver</code>	This interface provides support for resolving Namespaces
<code>XMLDocumentHandler</code>	This interface extends the <code>org.xml.sax.DocumentHandler</code> interface.
<code>XMLToken</code>	Basic interface for XMLToken

#### Classes

<code>AttrDecl</code>	This class hold information about each attribute declared in an attribute list in the Document Type Definition.
<code>Package</code> <code>oracle.xml.parser.v2</code>	This class implements the default behaviour for the <code>XMLDocumentHandler</code> interface.
<code>DOMParser</code>	This class implements an eXtensible Markup Language (XML) 1.0 parser according to the World Wide Web Consortium (W3C) recommendation.
<code>DTD</code>	Implements the DOM <code>DocumentType</code> interface and holds the Document Type Definition information for an XML document.
<code>ElementDecl</code>	This class represents an element declaration in a DTD.
<code>NodeFactory</code>	This class specifies methods to create various nodes of the DOM tree built during parsing.
<code>oraxsl</code>	The <code>oraxsl</code> class provides a command-line interface to applying stylesheets on multiple XML documents.
<code>SAXAttrlist</code>	This class implements the SAX <code>AttributeList</code> interface and also provides Namespace support.
<code>SAXParser</code>	This class implements an eXtensible Markup Language (XML) 1.0 SAX parser according to the World Wide Web Consortium (W3C) recommendation.
<code>XMLAttr</code>	This class implements the DOM <code>Attr</code> interface and holds information on each attribute of an element.
<code>XMLCDATA</code>	This class implements the DOM <code>CDATASection</code> interface.
<code>XMLComment</code>	This class implements the DOM <code>Comment</code> interface.
<code>XMLDocument</code>	This class implements the DOM <code>Document</code> interface, represents an entire XML document and serves the root of the Document Object Model tree.
<code>XMLDocumentFragment</code>	This class implements the DOM <code>DocumentFragment</code> interface.

---

**Class Summary**

---

<code>XMLElement</code>	This class implements the DOM <code>Element</code> interface.
<code>XMLEntityReference</code>	
<code>XMLNode</code>	Implements the DOM <code>Node</code> interface and serves as the primary datatype for the entire Document Object Model.
<code>XMLParser</code>	This class serves as a base class for the <code>DOMParser</code> and <code>SAXParser</code> classes.
<code>XMLPI</code>	This class implements the DOM Processing Instruction interface.
<code>XMLText</code>	This class implements the DOM Text interface.
<code>XMLTokenizer</code>	This class implements an eXtensible Markup Language (XML) 1.0 parser according to the World Wide Web Consortium (W3C) recommendation.
<code>XSLProcessor</code>	This class provides methods to transform an input XML document using a previously constructed <code>XSLStylesheet</code> .
<code>XSLStylesheet</code>	The class holds XSL stylesheet information such as templates, keys, variables, and attribute sets.
<b>Exceptions</b>	
<code>XMLParseException</code>	Indicates that a parsing exception occurred while processing an XML document
<code>XSLException</code>	Indicates that an exception occurred during XSL transformation

---

## AttrDecl

### Syntax

```
public class AttrDecl extends XMLNode implements  
oracle.xml.parser.v2.XMLConstants, java.io.Serializable
```

```
java.lang.Object  
|  
+--XMLNode  
|  
+--oracle.xml.parser.v2.AttrDecl
```

### All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.Node, java.io.Serializable,  
oracle.xml.parser.v2.XMLConstants
```

### Description

This class holds information about each attribute declared in an attribute list in the Document Type Definition.

---

### Member Summary

---

#### Fields

CDATA	AttType - StringType - CDATA
DEFAULT	Attribute presence - Default
ENTITIES	AttType - TokenizedType - Entities
ENTITY	AttType - TokenizedType - Entity
ENUMERATION	AttType - EnumeratedType - Enumeration
FIXED	Attribute presence - Fixed
ID	AttType - TokenizedType - ID
IDREF	AttType - TokenizedType - ID reference
IDREFS	AttType - TokenizedType - ID references
IMPLIED	Attribute presence - Implied
NMTOKEN	AttType - TokenizedType - Name token
NMTOKENS	AttType - TokenizedType - Name tokens

---

**Member Summary**

---

NOTATION	AttType - EnumeratedType - Notation
REQUIRED	Attribute presence - Required
<b>Methods</b>	
getAttrPresence()	Gets attribute presence
getAttrType()	Gets attribute type
getDefaultValue()	Gets attribute default value
getEnumerationValues()	Gets attribute values

---



---

**Inherited Member Summary**

---

**Fields inherited from class XMLNode**

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface oracle.xml.parser.v2.XMLConstants**

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface Node**

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

**Methods inherited from class XMLNode**

---

## Inherited Member Summary

---

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Fields

### CDATA

public static final int CDATA  
AttType - StringType - CDATA

### DEFAULT

public static final int DEFAULT  
Attribute presence - Default

### ENTITIES

public static final int ENTITIES  
AttType - TokenizedType - Entities

### ENTITY

public static final int ENTITY  
AttType - TokenizedType - Entity

### ENUMERATION

public static final int ENUMERATION  
AttType - EnumeratedType - Enumeration

**FIXED**

```
public static final int FIXED
Attribute presence - Fixed
```

**ID**

```
public static final int ID
AttType - TokenizedType - ID
```

**IDREF**

```
public static final int IDREF
AttType - TokenizedType - ID reference
```

**IDREFS**

```
public static final int IDREFS
AttType - TokenizedType - ID references
```

**IMPLIED**

```
public static final int IMPLIED
Attribute presence - Implied
```

**NMTOKEN**

```
public static final int NMTOKEN
AttType - TokenizedType - Name token
```

**NMTOKENS**

```
public static final int NMTOKENS
AttType - TokenizedType - Name tokens
```

**NOTATION**

```
public static final int NOTATION
AttType - EnumeratedType - Notation
```

**REQUIRED**

```
public static final int REQUIRED
Attribute presence - Required
```

## Methods

### **getAttrPresence()**

```
public int getAttrPresence()  
Gets attribute presence
```

#### **Returns**

The presence of the attribute

### **getAttrType()**

```
public int getAttrType()  
Gets attribute type
```

#### **Returns**

The type of the attribute

### **getDefaultValue()**

```
public java.lang.String getDefaultValue()  
Gets attribute default value
```

#### **Returns**

The default value of the attribute

### **getEnumerationValues()**

```
public java.util.Vector getEnumerationValues()  
Gets attribute values
```

#### **Returns**

The values of the attribute as an Enumeration



## DefaultXMLDocumentHandler

### Syntax

```
public class DefaultXMLDocumentHandler extends org.xml.sax.HandlerBase implements
XMLDocumentHandler
```

```
java.lang.Object
|
+--org.xml.sax.HandlerBase
|
+--oracle.xml.parser.v2.DefaultXMLDocumentHandler
```

### Direct Known Subclasses:

XMLTokenizer

### All Implemented Interfaces

org.xml.sax.DocumentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, XMLDocumentHandler

### Description

This class implements the default behaviour for the XMLDocumentHandler interface.

Application writers can extend this class when they need to implement only part of the interface

---

### Member Summary

---

#### Constructors

DefaultXMLDocumentHandler() Constructs a default document handler

#### Methods

cDATASection(char[], int, int)	Receive notification of a CDATA Section.
comment(String)	Receive notification of a comment.
endDoctype()	Receive notification of end of the DTD.
endElement(NSName)	Receive notification of the end of an element.
setDoctype(DTD)	Receive notification of DTD.
setTextDecl(String, String)	Receive notification of a Text XML Declaration.

---

### Member Summary

---

<code>setXMLDecl(String, String, String)</code>	Receive notification of an XML Declaration.
<code>startElement(NSName, SAXAttrList)</code>	Receive notification of the beginning of an element.

---

---

### Inherited Member Summary

---

Methods inherited from class `HandlerBase`

`characters(char[], int, int)`, `endDocument()`, `endElement(String)`, `error(SAXParseException)`, `fatalError(SAXParseException)`, `ignorableWhitespace(char[], int, int)`, `notationDecl(String, String, String)`, `processingInstruction(String, String)`, `resolveEntity(String, String)`, `setDocumentLocator(Locator)`, `startDocument()`, `startElement(String, AttributeList)`, `unparsedEntityDecl(String, String, String, String)`, `warning(SAXParseException)`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `DocumentHandler`

`characters(char[], int, int)`, `endDocument()`, `endElement(String)`, `ignorableWhitespace(char[], int, int)`, `processingInstruction(String, String)`, `setDocumentLocator(Locator)`, `startDocument()`, `startElement(String, AttributeList)`

Methods inherited from interface `EntityResolver`

`resolveEntity(String, String)`

Methods inherited from interface `DTDHandler`

`notationDecl(String, String, String)`, `unparsedEntityDecl(String, String, String, String)`

Methods inherited from interface `ErrorHandler`

`error(SAXParseException)`, `fatalError(SAXParseException)`, `warning(SAXParseException)`

---

## Constructor

### DefaultXMLDocumentHandler()

```
public DefaultXMLDocumentHandler()  
Constructs a default document handler
```

## Methods

### cDATASection(char[], int, int)

```
public void cDATASection(char[] ch, int start, int length)
```

Receive notification of a CDATA Section.

The Parser will invoke this method once for each CDATA Section found.

#### Specified By

cDATASection(char[ ], int, int) in interface XMLDocumentHandler

#### Parameters

`ch` - The CDATA section characters.

`start` - The start position in the character array.

`length` - The number of characters to use from the character array.

#### Throws

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

### comment(String)

```
public void comment(java.lang.String data)
```

Receive notification of a comment.

The Parser will invoke this method once for each comment found: note that comment may occur before or after the main document element.

#### Specified By

comment(String) in interface XMLDocumentHandler

#### Parameters

`data` - The comment data, or null if none was supplied.

#### Throws

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

## endDoctype()

```
public void endDoctype()
```

Receive notification of end of the DTD.

### Specified By

endDoctype() in interface XMLDocumentHandler

### Throws

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

## endElement(NSName)

```
public void endElement(NSName elem)
```

Receive notification of the end of an element. The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

By implementing this method instead of `org.xml.sax.DocumentHandler.endElement`, SAX Applications can get the Namespace support provided by `NSName`.

### Specified By

endElement(NSName) in interface XMLDocumentHandler

### Parameters

elem - NSName object

### Throws

org.xml.sax.SAXException - A SAXException could be thrown.

### See Also

org.xml.sax.DocumentHandler.endElement(String)

## setDoctype(DTD)

```
public void setDoctype(DTD dtd)
```

Receive notification of DTD. Sets the DTD

**Specified By**

setDoctype(DTD) in interface XMLDocumentHandler

**Throws**

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

**setTextDecl(String, String)**

```
public void setTextDecl(java.lang.String version, java.lang.String encoding)
```

Receive notification of a Text XML Declaration.

The Parser will invoke this method once for each text XML Decl

**Specified By**

setTextDecl(String, String) in interface XMLDocumentHandler

**Parameters**

version - The version number (or null, if not specified)

encoding - The encoding name

**Throws**

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

**setXMLDecl(String, String, String)**

```
public void setXMLDecl(java.lang.String version, java.lang.String standalone,  
java.lang.String encoding)
```

Receive notification of an XML Declaration.

The Parser will invoke this method once for XML Decl

**Specified By**

setXMLDecl(String, String, String) in interface XMLDocumentHandler

**Parameters**

version - The version number

standalone - The standalone value (or null, if not specified)

encoding - The encoding name (or null, if not specified)

### Throws

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

### startElement(NSName, SAXAttrList)

```
public void startElement(NSName elem, SAXAttrList attrlist)
```

Receive notification of the beginning of an element. The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

By implementing this method instead of

org.xml.sax.DocumentHandler.startElement, SAX Applications can get the Namespace support provided by NSName and SAXAttrList.

### Specified By

startElement(NSName, SAXAttrList) in interface XMLDocumentHandler

### Parameters

elem - NSName object

attrlist - SAXAttrList for the element

### Throws

org.xml.sax.SAXException - A SAXException could be thrown.

### See Also

org.xml.sax.DocumentHandler.startElement(String, AttributeList)

# DOMParser

## Syntax

```
public class DOMParser extends XMLParser implements
oracle.xml.parser.v2.XMLConstants
```

```
java.lang.Object
|
+--XMLParser
|
+--oracle.xml.parser.v2.DOMParser
```

## All Implemented Interfaces

```
oracle.xml.parser.v2.XMLConstants
```

## Description

This class implements an eXtensible Markup Language (XML) 1.0 parser according to the World Wide Web Consortium (W3C) recommendation to parse a XML document and build a DOM tree.

---

## Member Summary

---

### Constructors

DOMParser()                      Creates a new parser object.

### Methods

getDoctype()                      Get the DTD

getDocument()                      Gets the document

parseDTD(InputSource, String)      Parses the XML External DTD from given input source

parseDTD(InputStream, String)      Parses the XML External DTD from given input stream.

parseDTD(Reader, String)            Parses the XML External DTD from given input stream.

parseDTD(String, String)            Parses the XML External DTD from the URL indicated

parseDTD(URL, String)              Parses the XML External DTD document pointed to by the given URL and creates the corresponding XML document hierarchy.

setErrorStream(OutputStream)      Creates an output stream for the output of errors and warnings.

---

## Member Summary

---

<code>setErrorStream(OutputStream, String)</code>	Creates an output stream for the output of errors and warnings.
<code>setErrorStream(PrintWriter)</code>	Creates an output stream for the output of errors and warnings.
<code>setNodeFactory(NodeFactory)</code>	Set the node factory.
<code>showWarnings(boolean)</code>	Switch to determine whether to print warnings

---

---

## Inherited Member Summary

---

### Fields inherited from class `XMLParser`

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Fields inherited from interface `oracle.xml.parser.v2.XMLConstants`

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Methods inherited from class `XMLParser`

`getReleaseVersion()`, `getValidationMode()`, `parse(InputSource)`, `parse(InputStream)`, `parse(Reader)`, `parse(String)`, `parse(URL)`, `setBaseURL(URL)`, `setDoctype(DTD)`, `setLocale(Locale)`, `setPreserveWhitespace(boolean)`, `setValidationMode(boolean)`

### Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---



## Constructor

### DOMParser()

```
public DOMParser()  
Creates a new parser object.
```

## Methods

### getDoctype()

```
public DTD getDoctype()  
Get the DTD
```

#### Returns

The DTD

### getDocument()

```
public XMLDocument getDocument()  
Gets the document
```

#### Returns

The document being parsed

### parseDTD(InputSource, String)

```
public final void parseDTD(org.xml.sax.InputSource in, java.lang.String rootName)  
Parses the XML External DTD from given input source
```

#### Parameters

`in` - the `org.xml.sax.InputSouce` to parse

`rootName` - the element to be used as root Element

#### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

**parseDTD(InputStream, String)**

```
public final void parseDTD(java.io.InputStream in, java.lang.String rootName)
```

Parses the XML External DTD from given input stream. The base URL should be set for resolving external entities and DTD.

**Parameters**

`in` - the `InputStream` containing XML data to parse.

`rootName` - the element to be used as root `Element`

**Throws**

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

**See Also**

`setBaseURL(URL)`

**parseDTD(Reader, String)**

```
public final void parseDTD(java.io.Reader r, java.lang.String rootName)
```

Parses the XML External DTD from given input stream. The base URL should be set for resolving external entities and DTD.

**Parameters**

`r` - the `Reader` containing XML data to parse.

`rootName` - the element to be used as root `Element`

**Throws**

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

**See Also**

`setBaseURL(URL)`

## parseDTD(String, String)

```
public final void parseDTD(java.lang.String in, java.lang.String rootName)
```

Parses the XML External DTD from the URL indicated

### Parameters

`in` - the `String` containing the URL to parse from

`rootName` - the element to be used as root Element

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

## parseDTD(URL, String)

```
public final void parseDTD(java.net.URL url, java.lang.String rootName)
```

Parses the XML External DTD document pointed to by the given URL and creates the corresponding XML document hierarchy.

### Parameters

`url` - the url points to the XML document to parse.

`rootName` - the element to be used as root Element

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

## setErrorStream(OutputStream)

```
public final void setErrorStream(java.io.OutputStream out)
```

Creates an output stream for the output of errors and warnings. If an output stream for errors is not specified, the parser will use the standard error output stream `System.err` for outputting errors and warnings.

**Parameters**

`out` - The output stream to use for errors and warnings

**setErrorStream(OutputStream, String)**

```
public final void setErrorStream(java.io.OutputStream out, java.lang.String enc)
```

Creates an output stream for the output of errors and warnings. If an output stream for errors is not specified, the parser will use the standard error output stream `System.err` for outputting errors and warnings. Additionally, an `.exception` is thrown if the encoding specified is unsupported.

**Parameters**

`out` - The output stream to use for errors and warnings

`enc` - the encoding to use

**Throws**

`IOException` - if an unsupported encoding is specified

**setErrorStream(PrintWriter)**

```
public final void setErrorStream(java.io.PrintWriter out)
```

Creates an output stream for the output of errors and warnings. If an output stream for errors is not specified, the parser will use the standard error output stream `System.err` for outputting errors and warnings.

**Parameters**

`out` - The `PrintWriter` to use for errors and warnings

**setNodeFactory(NodeFactory)**

```
public void setNodeFactory(NodeFactory factory)
```

Set the node factory. Applications can extend the `NodeFactory` and register it through this method. The parser will then use the user supplied `NodeFactory` to create nodes of the DOM tree.

**Parameters**

`factory` - The `NodeFactory` to set

**Throws**

`XMLParseException` - if an invalid factory is set

**See Also**

NodeFactory

**showWarnings(boolean)**

```
public void showWarnings(boolean yes)
```

Switch to determine whether to print warnings

**Parameters**

`yes` - determines whether warnings should be shown

## DTD

### Syntax

```
public class DTD extends XMLNode implements org.w3c.dom.DocumentType,
java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.DTD
```

### All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.DocumentType, org.w3c.dom.Node,
java.io.Serializable, oracle.xml.parser.v2.XMLConstants
```

### Description

Implements the DOM DocumentType interface and holds the Document Type Definition information for an XML document.

---

### Member Summary

---

#### Methods

cloneNode(boolean)	Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
findElementDecl(String)	Finds an element declaration for the given tag name.
findEntity(String, boolean)	Finds a named entity in the DTD.
findNotation(String)	Retrieves the named notation from the DTD.
getChildNodes()	A <code>NodeList</code> that contains all children of this node.
getElementDecls()	A <code>NamedNodeMap</code> containing the element declarations in the DTD.
getEntities()	A <code>NamedNodeMap</code> containing the general entities, both external and internal, declared in the DTD.
getName()	Gets the name of the DTD; i.e., the name immediately following the <code>DOCTYPE</code> keyword.
getNotations()	A <code>NamedNodeMap</code> containing the notations declared in the DTD.
getPublicId()	Gets The public identifier associated with the DTD, if specified.

---

**Member Summary**


---

getSystemId()	Gets the system identifier associated with the DTD, if specified.
hasChildNodes()	This is a convenience method to allow easy determination of whether a node has any children.
printExternalDTD(OutputStream)	Writes the contents of this document to the given output stream.
printExternalDTD(OutputStream, String)	Writes the contents of the external DTD to the given output stream.
printExternalDTD(PrintWriter)	Writes the contents of this document to the given output stream.

---



---

**Inherited Member Summary**


---

**Fields inherited from class XMLNode**

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface Node**

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

**Fields inherited from interface oracle.xml.parser.v2.XMLConstants**

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Methods inherited from class XMLNode**

---

## Inherited Member Summary

---

appendChild(Node), getAttributes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface Node

appendChild(Node), getAttributes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Methods

### cloneNode(boolean)

```
public org.w3c.dom.Node cloneNode(boolean deep)
```

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (`parentNode` returns `null`). Cloning an `Element` copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child `Text` node. Cloning any other type of node simply returns a copy of this node.

### Specified By

`org.w3c.dom.Node.cloneNode(boolean)` in interface `org.w3c.dom.Node`

### Overrides

`cloneNode(boolean)` in class `XMLNode`

### Parameters

`deep` - If `true`, recursively clone the subtree under the specified node; if `false`, clone only the node itself (and its attributes, if it is an `Element`).

### Returns

The duplicate node.



**findElementDecl(String)**

```
public final ElementDecl findElementDecl(java.lang.String name)
```

Finds an element declaration for the given tag name.

**Parameters**

name - The tag name.

**Returns**

the element declaration object.

**findEntity(String, boolean)**

```
public final org.w3c.dom.Entity findEntity(java.lang.String n, boolean par)
```

Finds a named entity in the DTD.

**Parameters**

n - The name of the entity.

**Returns**

the specified `Entity` object; returns null if it is not found.

**findNotation(String)**

```
public final org.w3c.dom.Notation findNotation(java.lang.String name)
```

Retrieves the named notation from the DTD.

**Parameters**

name - The name of the notation.

**Returns**

the `Notation` object; returns null if it is not found.

**getChildNodes()**

```
public org.w3c.dom.NodeList getChildNodes()
```

A `NodeList` that contains all children of this node. If there are no children, this is a `NodeList` containing no nodes. The content of the returned `NodeList` is "live" in the sense that, for instance, changes to the children of the node object that it was created from are immediately reflected in the nodes returned by the `NodeList` accessors; it is not a static snapshot of the content of the node. This is true for every

`NodeList`, including the ones returned by the `getElementsByTagName` method.

**Specified By**

`org.w3c.dom.Node.getChildNodes()` in interface `org.w3c.dom.Node`

**Overrides**

`getChildNodes()` in class `XMLNode`

**Returns**

The children of this node

**getElementDecls()**

```
public org.w3c.dom.NamedNodeMap getElementDecls()
```

A `NamedNodeMap` containing the element declarations in the DTD. Every node in this map is an `ElementDecl` object.

**Returns**

The element declarations in the DTD The DOM Level 1 does not support editing `elementdecls`, therefore `elementdecls` cannot be altered in any way.

**getEntities()**

```
public org.w3c.dom.NamedNodeMap getEntities()
```

A `NamedNodeMap` containing the general entities, both external and internal, declared in the DTD. Duplicates are discarded. For example in:`<!DOCTYPE ex SYSTEM "ex.dtd" [ <!ENTITY foo "foo"> <!ENTITY bar "bar"> <!ENTITY % baz "baz">]> <ex/>` the interface provides access to `foo` and `bar` but not `baz`. Every node in this map also implements the `Entity` interface. The DOM Level 1 does not support editing entities, therefore `entities` cannot be altered in any way.

**Specified By**

`org.w3c.dom.DocumentType.getEntities()` in interface `org.w3c.dom.DocumentType`

**Returns**

The entities declared in the DTD

## getName()

```
public java.lang.String getName()
```

Gets the name of the DTD; i.e., the name immediately following the `DOCTYPE` keyword.

### Specified By

`org.w3c.dom.DocumentType.getName()` in interface `org.w3c.dom.DocumentType`

### Returns

Name of the DTD

## getNotations()

```
public org.w3c.dom.NamedNodeMap getNotations()
```

A `NamedNodeMap` containing the notations declared in the DTD. Duplicates are discarded. Every node in this map also implements the `Notation` interface. The DOM Level 1 does not support editing notations, therefore notations cannot be altered in any way.

### Specified By

`org.w3c.dom.DocumentType.getNotations()` in interface `org.w3c.dom.DocumentType`

### Returns

The notations declared in the DTD

## getPublicId()

```
public java.lang.String getPublicId()
```

Gets The public identifier associated with the DTD, if specified. If the public identifier was not specified, this is `null`.

### Returns

the public identifier associated with the DTD

## getSystemId()

```
public java.lang.String getSystemId()
```

Gets the system identifier associated with the DTD, if specified. If the system identifier was not specified, this is `null`.

**Returns**

the system identifier associated with the DTD

**hasChildNodes()**

```
public boolean hasChildNodes()
```

This is a convenience method to allow easy determination of whether a node has any children. return false always, as DTD cannot have any overrides method in XMLNode

**Specified By**

org.w3c.dom.Node.hasChildNodes() in interface org.w3c.dom.Node

**Overrides**

hasChildNodes() in class XMLNode

**Returns**

false as DTD node can not have any children,

**printExternalDTD(OutputStream)**

```
public void printExternalDTD(java.io.OutputStream out)
```

Writes the contents of this document to the given output stream.

**Parameters**

out - OutputStream to write to

**Throws**

IOException - if an error occurs

**printExternalDTD(OutputStream, String)**

```
public void printExternalDTD(java.io.OutputStream out, java.lang.String enc)
```

Writes the contents of the external DTD to the given output stream.

**Parameters**

out - OutputStream to write to

enc - Encoding to use for the output

**Throws**

`IOException` - if an invalid encoding was specified or if any other error occurs

**printExternalDTD(PrintWriter)**

```
public void printExternalDTD(java.io.PrintWriter out)
```

Writes the contents of this document to the given output stream.

**Parameters**

`out` - `PrintWriter` to write to

**Throws**

`IOException` - if an error occurs

## ElementDecl

### Syntax

```
public class ElementDecl extends XMLNode implements java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
|
+--oracle.xml.parser.v2.ElementDecl
```

### All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.Node, java.io.Serializable,
oracle.xml.parser.v2.XMLConstants
```

### Description

This class represents an element declaration in a DTD.

---

### Member Summary

---

#### Fields

ANY	Element content type - Children can be any element
ASTERISK	ContentModelParseTreeNode type - "*" node (has one children)
COMMA	ContentModelParseTreeNode type - "," node (has two children)
ELEMENT	ContentModelParseTreeNode type - 'leaf' node (has no children)
ELEMENTS	Element content type - Children can be elements as per Content Model
EMPTY	Element content type - No Children
MIXED	Element content type - Children can be PCDATA & elements as per Content Model
OR	ContentModelParseTreeNode type - " " node (has two children)
PLUS	ContentModelParseTreeNode type - "+" node (has one children)
QMARK	ContentModelParseTreeNode type - "?" node (has one children)

#### Methods

expectedElements(Element)	Returns vector of element names that can be appended to the element.
findAttrDecl(String)	Gets an attribute declaration object or null if not found

---

**Member Summary**


---

getAttrDecls()	Gets an enumeration of attribute declarations
getContentElements()	Returns Vector of elements that can be appended to this element
getContentType()	Returns content model of element
getParseTree()	Returns the root node of Content Model Parse Tree.
validateContent(Element)	Validates the content of a element node.

---



---

**Inherited Member Summary**


---

**Fields inherited from class XMLNode**

AMP, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, PERCENT, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface Node**

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

**Fields inherited from interface oracle.xml.parser.v2.XMLConstants**

AMP, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, PERCENT, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Methods inherited from class XMLNode**

---

## Inherited Member Summary

---

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Fields

### ANY

public static final byte ANY  
Element content type - Children can be any element

### ASTERISK

public static final int ASTERISK  
ContentModelParseTreeNode type - "\*" node (has one children)

### COMMA

public static final int COMMA  
ContentModelParseTreeNode type - "," node (has two children)

### ELEMENT

public static final int ELEMENT  
ContentModelParseTreeNode type - 'leaf' node (has no children)

### ELEMENTS

public static final byte ELEMENTS  
Element content type - Children can be elements as per Content Model



**EMPTY**

```
public static final byte EMPTY
Element content type - No Children
```

**MIXED**

```
public static final byte MIXED
Element content type - Children can be PCDATA & elements as per Content Model
```

**OR**

```
public static final int OR
ContentModelParseTreeNode type - "|" node (has two children)
```

**PLUS**

```
public static final int PLUS
ContentModelParseTreeNode type - "+" node (has one children)
```

**QMARK**

```
public static final int QMARK
ContentModelParseTreeNode type - "?" node (has one children)
```

**Methods****expectedElements(Element)**

```
public java.util.Vector expectedElements(org.w3c.dom.Element e)
Returns vector of element names that can be appended to the element.
```

**Parameters**

e - Element

**Returns**

Vector of names

**findAttrDecl(String)**

```
public final AttrDecl findAttrDecl(java.lang.String name)
Gets an attribute declaration object or null if not found
```

### Parameters

name - Attribute declaration to find

### Returns

The `AttrDecl` object, or null, if it was not found

### getAttrDecls()

```
public org.w3c.dom.NamedNodeMap getAttrDecls()
```

Gets an enumeration of attribute declarations

### Returns

An enumeration of attribute declarations

### getContentElements()

```
public final java.util.Vector getContentElements()
```

Returns `Vector` of elements that can be appended to this element

### Returns

The `Vector` containing the element names.

### getContentType()

```
public int getContentType()
```

Returns content model of element

### Returns

The `type` of the element declaration.

### getParseTree()

```
public final org.w3c.dom.Node getParseTree()
```

Returns the root node of Content Model Parse Tree. `Node.getFirstChild()` and `Node.getLastChild()` return the the parse tree branches.

`Node.getNodeType()` and `Node.getNodeName()` return the the parse tree node type and name.

### Returns

The `Node` containing the Content Model parse tree root node.

**validateContent(Element)**

```
public boolean validateContent(org.w3c.dom.Element e)
```

Validates the content of a element node.

**Returns**

True if valid, else false

## NodeFactory

### Syntax

```
public class NodeFactory extends java.lang.Object implements  
java.io.Serializable
```

```
java.lang.Object  
|  
+--oracle.xml.parser.v2.NodeFactory
```

### All Implemented Interfaces

```
java.io.Serializable
```

### Description

This class specifies methods to create various nodes of the DOM tree built during parsing. Applications can override these methods to create their own custom classes to be added to the DOM tree while parsing. Applications have to register their own NodeFactory using the XMLParser's setNodeFactory() method. If a null pointer is returned by these methods, then the node will not be added to the DOM tree.

### See Also

```
setNodeFactory(NodeFactory)
```

---

## Member Summary

---

### Constructors

```
NodeFactory()
```

### Methods

<pre>createAttribute(String, String)</pre>	Creates an attribute node with the specified tag, and text.
<pre>createCDATASection(String)</pre>	Creates a CDATA node with the specified text.
<pre>createComment(String)</pre>	Creates a comment node with the specified text.
<pre>createDocument()</pre>	Creates a document node.
<pre>createElement(String)</pre>	Creates an Element node with the specified tag.
<pre>createProcessingInstruction(String, String)</pre>	Creates a PI node with the specified tag, and text.

---

**Member Summary**

<code>createTextNode(String)</code>	Creates a text node with the specified text.
-------------------------------------	--

---

---

**Inherited Member Summary**

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

**Constructor****NodeFactory()**

```
public NodeFactory()
```

**Methods****createAttribute(String, String)**

```
public XMLAttr createAttribute(java.lang.String tag, java.lang.String text)
```

Creates an attribute node with the specified tag, and text.

**Parameters**

`tag` - The name of the node.

`text` - The text associated with the node.

**Returns**

The created attribute node.

**createCDATASection(String)**

```
public XMLCDATA createCDATASection(java.lang.String text)
```

Creates a CDATA node with the specified text.

**Parameters**

`text` - The text associated with the node.

**Returns**

The created CDATA node.

**createComment(String)**

```
public XMLComment createComment(java.lang.String text)
```

Creates a comment node with the specified text.

**Parameters**

text - The text associated with the node.

**Returns**

The created comment node.

**createDocument()**

```
public XMLDocument createDocument()
```

Creates a document node. This method cannot return a null pointer.

**Returns**

The created element.

**createElement(String)**

```
public XMLElement createElement(java.lang.String tag)
```

Creates an Element node with the specified tag.

**Parameters**

tag - The name of the element.

**Returns**

The created element.

**createProcessingInstruction(String, String)**

```
public XMLPI createProcessingInstruction(java.lang.String tag, java.lang.String text)
```

Creates a PI node with the specified tag, and text.

**Parameters**

tag - The name of the node.

`text` - The text associated with the node.

**Returns**

The created PI node.

**createTextNode(String)**

```
public XMLText createTextNode(java.lang.String text)
```

Creates a text node with the specified text.

**Parameters**

`text` - The text associated with the node.

**Returns**

The created text node.

## NSName

### Syntax

```
public interface NSName
```

### All Known Implementing Classes:

*XMLAttr*, *XMLElement*

### Description

This interface provides Namespace support for Element and Attr names

---

### Member Summary

---

#### Methods

<code>getExpandedName()</code>	Get the fully resolved name for this name
<code>getLocalName()</code>	Get the local name for this name
<code>getNamespace()</code>	Get the resolved Namespace for this name
<code>getPrefix()</code>	Get the prefix for this name
<code>getQualifiedName()</code>	Get the qualified name

---

### Methods

#### **getExpandedName()**

```
public java.lang.String getExpandedName()  
Get the fully resolved name for this name
```

#### **Returns**

The fully resolved name

#### **getLocalName()**

```
public java.lang.String getLocalName()  
Get the local name for this name
```



**Returns**

The local name

**getNamespace()**

```
public java.lang.String getNamespace()  
Get the resolved Namespace for this name
```

**Returns**

The resolved Namespace

**getPrefix()**

```
public java.lang.String getPrefix()  
Get the prefix for this name
```

**Returns**

The prefix

**getQualifiedName()**

```
public java.lang.String getQualifiedName()  
Get the qualified name
```

**Returns**

The qualified name

## NSResolver

### Syntax

```
public interface NSResolver
```

### All Known Implementing Classes

`XMLElement`

### Description

This interface provides support for resolving Namespaces

---

### Member Summary

---

#### Methods

<code>resolveNamespacePrefix(String)</code>	Find the namespace definition in scope for a given namespace prefix
---	---

---

### Methods

#### `resolveNamespacePrefix(String)`

```
public java.lang.String resolveNamespacePrefix(java.lang.String prefix)
```

Find the namespace definition in scope for a given namespace prefix

#### Parameters

`prefix` - Namespace prefix to be resolved

#### Returns

the resolved Namespace (null, if prefix could not be resolved)

# oraxsl

## Syntax

```
public class oraxsl extends java.lang.Object
```

```
java.lang.Object
|
+--oracle.xml.parser.v2.Oraxsl
```

## Description

The `oraxsl` class provides a command-line interface to applying stylesheets on multiple XML documents. It accepts a number of command-line options that dictate how it should behave. The following is its invocation syntax:

```
java oraxsl options* source? stylesheet? result?
-w                               Show warnings
-e <error log>                   A file to write errors to
-l <xml file list>               List of files to transform
-d <directory>                  Directory with files to transform
-x <source extension>           Extensions to exclude
-i <source extension>           Extensions to include
-s <stylesheet>                 Stylesheet to use
-r <result extension>           Extension to use for results
-o <result extension>           Directory to place results
-p <param list>                 List of Params
-t <# of threads>               Number of threads to use
-v                               Verbose mode
```

---

## Member Summary

### Constructors

`oraxsl()`

### Methods

`main(String[])`                      Invokes the oraxsl driver

---



---

## Inherited Member Summary

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

## Constructor

### oraxsl()

```
public oraxsl()
```

## Methods

### main(String[])

```
public static void main(java.lang.String[] args)
```

Invokes the oraxsl driver

#### Parameters

`args` - command line arguments

## SAXAttrList

### Syntax

```
public class SAXAttrList extends java.lang.Object implements
org.xml.sax.AttributeList
```

```
java.lang.Object
|
+--oracle.xml.parser.v2.SAXAttrList
```

### All Implemented Interfaces

```
org.xml.sax.AttributeList
```

### Description

This class implements the SAX `AttributeList` interface and also provides Namespace support. Applications that require Namespace support can explicitly cast any attribute list returned by an Oracle parser class to `SAXAttrList` and use the methods described here.

---

### Member Summary

---

#### Methods

<code>getExpandedName(int)</code>	Get the expanded name for an attribute in the list (by position)
<code>getLength()</code>	Return the number of attributes in this list.
<code>getLocalName(int)</code>	Get the local name for an attribute in the list (by position)
<code>getName(int)</code>	Return the name of an attribute in this list (by position).
<code>getNamespace(int)</code>	Get the resolved namespace for an attribute in the list (by position)
<code>getPrefix(int)</code>	Get the namespace prefix for an attribute in the list (by position)
<code>getQualifiedName(int)</code>	Get the qualified name for an attribute in the list (by position)
<code>getType(int)</code>	
<code>getType(String)</code>	Return the type of an attribute in the list (by name).
<code>getValue(int)</code>	Return the value of an attribute in the list (by position).
<code>getValue(String)</code>	Return the value of an attribute in the list (by name).

---

---

## Inherited Member Summary

---

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

## Methods

### `getExpandedName(int)`

```
public java.lang.String getExpandedName(int i)
```

Get the expanded name for an attribute in the list (by position)

#### Parameters

`i` - The index of the attribute in the list.

#### Returns

The expanded name for the attribute

### `getLength()`

```
public int getLength()
```

Return the number of attributes in this list.

The SAX parser may provide attributes in any arbitrary order, regardless of the order in which they were declared or specified. The number of attributes may be zero.

#### Specified By

`org.xml.sax.AttributeList.getLength()` in interface `org.xml.sax.AttributeList`

#### Returns

The number of attributes in the list.

### `getLocalName(int)`

```
public java.lang.String getLocalName(int i)
```

Get the local name for an attribute in the list (by position)

**Parameters**

*i* - The index of the attribute in the list.

**Returns**

The local name for the attribute

**getName(int)**

```
public java.lang.String getName(int i)
```

Return the name of an attribute in this list (by position).

The names must be unique the SAX parser shall not include the same attribute twice. Attributes without values (those declared #IMPLIED without a value specified in the start tag) will be omitted from the list.

If the attribute name has a namespace prefix, the prefix will still be attached.

**Specified By**

org.xml.sax.AttributeList.getName(int) in interface org.xml.sax.AttributeList

**Parameters**

*i* - The index of the attribute in the list (starting at 0).

**Returns**

The name of the indexed attribute, or null if the index is out of range.

**See Also**

getLength()

**getNamespace(int)**

```
public java.lang.String getNamespace(int i)
```

Get the resolved namespace for an attribute in the list (by position)

**Parameters**

*i* - The index of the attribute in the list.

**Returns**

The resolved namespace for the attribute

**getPrefix(int)**

```
public java.lang.String getPrefix(int i)
```

Get the namespace prefix for an attribute in the list (by position)

**Parameters**

*i* - The index of the attribute in the list.

**Returns**

The namespace prefix for the attribute

**getQualifiedName(int)**

```
public java.lang.String getQualifiedName(int i)
```

Get the qualified name for an attribute in the list (by position)

**Parameters**

*i* - The index of the attribute in the list.

**Returns**

The qualified name for the attribute

**getType(int)**

```
public java.lang.String getType(int i)
```

**Specified By**

`org.xml.sax.AttributeList.getType(int)` in interface `org.xml.sax.AttributeList`

**getType(String)**

```
public java.lang.String getType(java.lang.String s)
```

Return the type of an attribute in the list (by name).

The return value is the same as the return value for `getType(int)`.

If the attribute name has a namespace prefix in the document, the application must include the prefix here.



**Specified By**

`org.xml.sax.AttributeList.getType(String)` in interface `org.xml.sax.AttributeList`

**Parameters**

`name` - The name of the attribute.

**Returns**

The attribute type as a string, or null if no such attribute exists.

**See Also**

`getType(int)`

**getValue(int)**

```
public java.lang.String getValue(int i)
```

Return the value of an attribute in the list (by position).

If the attribute value is a list of tokens (IDREFS, ENTITIES, or NMTOKENS), the tokens will be concatenated into a single string separated by whitespace.

**Specified By**

`org.xml.sax.AttributeList.getValue(int)` in interface `org.xml.sax.AttributeList`

**Parameters**

`i` - The index of the attribute in the list (starting at 0).

**Returns**

The attribute value as a string, or null if the index is out of range.

**See Also**

`getLength()`, `getValue(String)`

**getValue(String)**

```
public java.lang.String getValue(java.lang.String s)
```

Return the value of an attribute in the list (by name).

The return value is the same as the return value for `getValue(int)`.

If the attribute name has a namespace prefix in the document, the application must include the prefix here.

**Specified By**

`org.xml.sax.AttributeList.getValue(String)` in interface `org.xml.sax.AttributeList`

**Parameters**

`i` - The index of the attribute in the list.

**Returns**

The attribute value as a string, or null if no such attribute exists.

**See Also**

`getValue(int)`

# SAXParser

## Syntax

public class SAXParser extends XMLParser implements org.xml.sax.Parser ,  
oracle.xml.parser.v2.XMLConstants

```

java.lang.Object
|
+--XMLParser
   |
   +--oracle.xml.parser.v2.SAXParser

```

## All Implemented Interfaces

org.xml.sax.Parser , oracle.xml.parser.v2.XMLConstants

## Description

This class implements an eXtensible Markup Language (XML) 1.0 SAX parser according to the World Wide Web Consortium (W3C) recommendation. Applications can register a SAX handler to receive notification of various parser events.

---

## Member Summary

---

### Constructors

SAXParser()                      Creates a new parser object.

### Methods

setDocumentHandler(DocumentHandler)      SAX applications can use this to register a new document event handler.

setDTDHandler(DTDHandler)              SAX applications can use this to register a new DTD event handler.

setEntityResolver(EntityResolver)      SAX applications can use this to register a new entity resolver

setErrorHandler(ErrorHandler)          SAX applications can use this to register a new error event handler.

---



---

## Inherited Member Summary

---

Fields inherited from class XMLParser

---

## Inherited Member Summary

---

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Methods inherited from class XMLParser

getReleaseVersion(), getValidationMode(), parse(InputSource), parse(InputStream), parse(Reader), parse(String), parse(URL), setBaseURL(URL), setDoctype(DTD), setLocale(Locale), setPreserveWhitespace(boolean), setValidationMode(boolean)

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface Parser

parse(InputSource), parse(String), setLocale(Locale)

---

## Constructor

### SAXParser()

```
public SAXParser()  
Creates a new parser object.
```

## Methods

### setDocumentHandler(DocumentHandler)

```
public void setDocumentHandler(org.xml.sax.DocumentHandler handler)
```

SAX applications can use this to register a new document event handler.

**Specified By**

`org.xml.sax.Parser.setDocumentHandler(DocumentHandler)` in interface `org.xml.sax.Parser`

**Parameters**

`handler` - `DocumentHandler` being registered

**See Also**

`org.xml.sax.Parser.setDocumentHandler(DocumentHandler)`, `org.xml.sax.DocumentHandler`

**setDTDHandler(DTDHandler)**

```
public void setDTDHandler(org.xml.sax.DTDHandler handler)
```

SAX applications can use this to register a new DTD event handler.

**Specified By**

`org.xml.sax.Parser.setDTDHandler(DTDHandler)` in interface `org.xml.sax.Parser`

**Parameters**

`handler` - `DTDHandler` being registered

**See Also**

`org.xml.sax.Parser.setDTDHandler(DTDHandler)`, `org.xml.sax.DTDHandler`

**setEntityResolver(EntityResolver)**

```
public void setEntityResolver(org.xml.sax.EntityResolver resolver)
```

SAX applications can use this to register a new entity resolver

**Specified By**

`org.xml.sax.Parser.setEntityResolver(EntityResolver)` in interface `org.xml.sax.Parser`

**Parameters**

`resolver` - `EntityResolver` being registered

**See Also**

`org.xml.sax.Parser.setEntityResolver(EntityResolver)`, `org.xml.sax.DTDHandler`

## **setErrorHandler(ErrorHandler)**

```
public void setErrorHandler(org.xml.sax.ErrorHandler handler)
```

SAX applications can use this to register a new error event handler. This replaces any previous setting for error handling.

### **Specified By**

`org.xml.sax.Parser.setErrorHandler(ErrorHandler)` in interface `org.xml.sax.Parser`

### **Parameters**

`handler` - `ErrorHandler` being registered

### **See Also**

`org.xml.sax.Parser.setErrorHandler(ErrorHandler)`, `org.xml.sax.ErrorHandler`

# XMLAttr

## Syntax

public class XMLAttr extends XMLNode implements org.w3c.dom.Attr, NSName, java.io.Serializable

```

java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.XMLAttr

```

## All Implemented Interfaces

org.w3c.dom.Attr, java.lang.Cloneable, org.w3c.dom.Node, NSName, java.io.Serializable, oracle.xml.parser.v2.XMLConstants

## Description

This class implements the DOM Attr interface and holds information on each attribute of an element.

## See Also

org.w3c.dom.Attr, NodeFactory, setNodeFactory(NodeFactory)

---

## Member Summary

---

### Constructors

XMLAttr(String, String) Construct attribute with given name and value.

XMLAttr(String, String, String, String) Namespace support

### Methods

cloneNode(boolean) Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.

getExpandedName() Get the fully resolved Name for this attribute

getLocalName() Get the local Name for this attribute

getName() Gets the attribute name.

getNamespace() Get the resolved Namespace for this attribute

getNodeValue() Gets the value of this node, depending on its type

---

**Member Summary**

---

getParentNode()	Gets the parent of this node.
getPrefix()	Get the namespace prefix for this attribute
getQualifiedName()	Gets the qualified name for this attribute
getSpecified()	Returns true if the attribute was specified explicitly in the element
getValue()	Gets the attribute value.
setNodeValue(String)	Sets the value of this node, depending on its type
setValue(String)	Sets the value.

---

---

**Inherited Member Summary**

---

## Fields inherited from class XMLNode

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREf, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMtOKEN, cNMtOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

## Fields inherited from interface Node

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

## Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREf, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMtOKEN, cNMtOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

## Methods inherited from class XMLNode



---

**Inherited Member Summary**

---

appendChild(Node), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeType(), getOwnerDocument(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), transformNode(XSLStylesheet), valueOf(String, NSResolver)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface Node

appendChild(Node), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeType(), getOwnerDocument(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node)

---

**Constructor****XMLAttr(String, String)**

```
public XMLAttr(java.lang.String n, java.lang.String v)
```

Construct attribute with given name and value.

**Parameters**

n - Name of the attribute

v - Value of the attribute

**XMLAttr(String, String, String, String)**

```
public XMLAttr(java.lang.String name, java.lang.String prefix, java.lang.String ns, java.lang.String v)
```

Namespace support

**Methods****cloneNode(boolean)**

```
public org.w3c.dom.Node cloneNode(boolean deep)
```

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (`parentNode` returns null). Cloning an `Element` copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any

text it contains unless it is a deep clone, since the text is contained in a child `Text` node. Cloning any other type of node simply returns a copy of this node.

**Specified By**

`org.w3c.dom.Node.cloneNode(boolean)` in interface `org.w3c.dom.Node`

**Overrides**

`cloneNode(boolean)` in class `XMLNode`

**Parameters**

`deep` - If `true`, recursively clone the subtree under the specified node; if `false`, clone only the node itself (and its attributes, if it is an `Element`).

**Returns**

The duplicate node.

**getExpandedName()**

```
public java.lang.String getExpandedName()  
Get the fully resolved Name for this attribute
```

**Specified By**

`getExpandedName()` in interface `NSName`

**Returns**

the fully resolved Name

**getLocalName()**

```
public java.lang.String getLocalName()  
Get the local Name for this attribute
```

**Specified By**

`getLocalName()` in interface `NSName`

**Returns**

the local Name

**getName()**

```
public java.lang.String getName()
```

Gets the attribute name.

**Specified By**

org.w3c.dom.Attr.getName() in interface org.w3c.dom.Attr

**Returns**

attribute name

**getNamespace()**

```
public java.lang.String getNamespace()
```

Get the resolved Namespace for this attribute

**Specified By**

getNamespace() in interface NSName

**Returns**

the resolved Namespace

**getNodeValue()**

```
public java.lang.String getNodeValue()
```

Gets the value of this node, depending on its type

**Specified By**

org.w3c.dom.Node.getNodeValue() in interface org.w3c.dom.Node

**Overrides**

getNodeValue() in class XMLNode

**Returns**

Value of this node

**Throws**

`org.w3c.dom.DOMException - NO_MODIFICATION_ALLOWED_ERR`: Raised when the node is readonly. `DOMSTRING_SIZE_ERR`: Raised when it would return more characters than fit in a `DOMString` variable on the implementation platform.

**getParentNode()**

```
public org.w3c.dom.Node getParentNode()
```

Gets the parent of this node. All nodes, except `Document`, `DocumentFragment`, and `Attr` may have a parent. However, if a node has just been created and not yet added to the tree, or if it has been removed from the tree, this is `null`.

**Specified By**

`org.w3c.dom.Node.getParentNode()` in interface `org.w3c.dom.Node`

**Overrides**

`getParentNode()` in class `XMLNode`

**Returns**

The parent of this node

**getPrefix()**

```
public java.lang.String getPrefix()
```

Get the namespace prefix for this attribute

**Specified By**

`getPrefix()` in interface `NSName`

**Returns**

the namespace prefix

**getQualifiedName()**

```
public java.lang.String getQualifiedName()
```

Gets the qualified name for this attribute

**Specified By**

`getQualifiedName()` in interface `NSName`

**Returns**

the qualified name

**getSpecified()**

```
public boolean getSpecified()
```

Returns true if the attribute was specified explicitly in the element

**Specified By**

`org.w3c.dom.Attr.getSpecified()` in interface `org.w3c.dom.Attr`

**Returns**

true, if the attribute was specified explicitly, false, if it was not

**getValue()**

```
public java.lang.String getValue()
```

Gets the attribute value.

**Specified By**

`org.w3c.dom.Attr.getValue()` in interface `org.w3c.dom.Attr`

**Returns**

attribute value

**setNodeValue(String)**

```
public void setNodeValue(java.lang.String nodeValue)
```

Sets the value of this node, depending on its type

**Specified By**

`org.w3c.dom.Node.setNodeValue(String)` in interface `org.w3c.dom.Node`

**Overrides**

`setNodeValue(String)` in class `XMLNode`

**Throws**

`org.w3c.dom.DOMException - NO_MODIFICATION_ALLOWED_ERR`: Raised when the node is readonly. `DOMSTRING_SIZE_ERR`: Raised when it would return more characters than fit in a `DOMString` variable on the implementation platform.

**setValue(String)**

```
public void setValue(java.lang.String arg)
```

Sets the value.

**Specified By**

`org.w3c.dom.Attr.setValue(String)` in interface `org.w3c.dom.Attr`

**Parameters**

`arg` - Value to set

# XMLCDATA

## Syntax

```
public class XMLCDATA extends XMLText implements org.w3c.dom.CDATASection,
java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
    |
    +--oracle.xml.parser.v2.CharData
        |
        +--XMLText
            |
            +--oracle.xml.parser.v2.XMLCDATA
```

## All Implemented Interfaces

org.w3c.dom.CDATASection, org.w3c.dom.CharacterData, java.lang.Cloneable, org.w3c.dom.Node, java.io.Serializable, org.w3c.dom.Text, oracle.xml.parser.v2.XMLConstants

## Description

This class implements the DOM CDATASection interface.

## See Also

org.w3c.dom.CDATASection, NodeFactory, setNodeFactory(NodeFactory)

---

## Member Summary

### Constructors

XMLCDATA(String)	Creates a CDATA node having the given name and text.
------------------	--

---

## Inherited Member Summary

Fields inherited from class XMLNode

---

## Inherited Member Summary

---

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREf, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNCDATA, cNMTOkEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Fields inherited from interface Node

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

### Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREf, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNCDATA, cNMTOkEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Methods inherited from class XMLText

appendData, deleteData, getData, getLength, getNodeValue(), insertData, replaceData, setData, setNodeValue, splitText(int), substringData

### Methods inherited from class oracle.xml.parser.v2.CharData

appendData, deleteData, getData, getLength, insertData, replaceData, setData, setNodeValue, substringData

### Methods inherited from class XMLNode

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeType(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), transformNode(XSLStylesheet), valueOf(String, NSResolver)

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface Text



---

## Inherited Member Summary

---

splitText(int)

Methods inherited from interface CharacterData

appendData(String), deleteData(int, int), getData(), getLength(), insertData(int, String), replaceData(int, int, String), setData(String), substringData(int, int)

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Constructor

### XMLCDATA(String)

```
public XMLCDATA(java.lang.String text)
```

Creates a CDATA node having the given name and text.

#### Parameters

text - Text of the node

## XMLComment

### Syntax

public class XMLComment extends oracle.xml.parser.v2.CharData implements org.w3c.dom.Comment, java.io.Serializable

```
java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.CharData
      |
      +--oracle.xml.parser.v2.XMLComment
```

### All Implemented Interfaces

org.w3c.dom.CharacterData, java.lang.Cloneable, org.w3c.dom.Comment, org.w3c.dom.Node, java.io.Serializable, oracle.xml.parser.v2.XMLConstants

### Description

This class implements the DOM Comment interface.

### See Also

org.w3c.dom.Comment, NodeFactory, setNodeFactory(NodeFactory)

---

### Member Summary

---

#### Constructors

XMLComment(String)	Creates a new Comment node.
--------------------	-----------------------------

---

---

### Inherited Member Summary

---

Fields inherited from class XMLNode

---

## Inherited Member Summary

---

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Fields inherited from interface Node

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

### Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Methods inherited from class oracle.xml.parser.v2.CharData

appendData, deleteData, getData, getLength, insertData, replaceData, setData, setNodeValue, substringData

### Methods inherited from class XMLNode

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), transformNode(XSLStylesheet), valueOf(String, NSResolver)

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface CharacterData

appendData(String), deleteData(int, int), getData(), getLength(), insertData(int, String), replaceData(int, int, String), setData(String), substringData(int, int)

### Methods inherited from interface Node

---

## Inherited Member Summary

---

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Constructor

### XMLComment(String)

```
public XMLComment(java.lang.String text)
```

Creates a new Comment node.

#### Parameters

text - Text of the comment node

## XMLDocument

### Syntax

```
public class XMLDocument extends XMLNode implements org.w3c.dom.Document ,
java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
|
+--oracle.xml.parser.v2.XMLDocument
```

### All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.Document, org.w3c.dom.Node ,
java.io.Serializable, oracle.xml.parser.v2.XMLConstants
```

### Description

This class implements the DOM Document interface, represents an entire XML document and serves the root of the Document Object Model tree. Each XML tag can either represent a node or a leaf of this tree.

According to the XML specification, the root of the tree consists of any combination of comments and processing instructions, but only one root element. A helper method `getElement` is provided as a short cut to finding the root element.

---

### Member Summary

---

#### Constructors

`XMLDocument()`                      Creates an empty document.

#### Methods

`cloneNode(boolean)`                Returns a duplicate of this document node.

`createAttribute(String)`            Creates an `Attr` of the given name.

`createCDATASection(String)`        Creates a `CDATASection` node whose value is the specified string.

`createComment(String)`              Creates a `Comment` node given the specified string.

`createDocumentFragment()`          Creates an empty `DocumentFragment` object.

`createElement(String)`              Creates an element of the type specified.

---

**Member Summary**

---

<code>createEntityReference(String)</code>	Creates an <code>EntityReference</code> object.
<code>createProcessingInstruction(String, String)</code>	Creates a <code>ProcessingInstruction</code> node given the specified name and data strings.
<code>createTextNode(String)</code>	Creates a <code>Text</code> node given the specified string.
<code>expectedElements(Element)</code>	Returns vector of element names that can be appended to the element.
<code>getDoctype()</code>	The <code>Document Type Declaration (DTD)</code> associated with this document.
<code>getDocumentElement()</code>	This is a convenience attribute that allows direct access to the child node that is the root element of the document.
<code>getElementsByTagName(String)</code>	Returns a <code>NodeList</code> of all the <code>Elements</code> with a given tag name in the order in which they would be encountered in a preorder traversal of the <code>Document</code> tree.
<code>getEncoding()</code>	Retrieves the character encoding information.
<code>getImplementation()</code>	The <code>DOMImplementation</code> object that handles this document.
<code>getOwnerDocument()</code>	The <code>Document</code> object associated with this node.
<code>getStandalone()</code>	Retrieves the standalone information.
<code>getVersion()</code>	Retrieves the version information.
<code>print(OutputStream)</code>	Writes the contents of this document to the given output stream.
<code>print(OutputStream, String)</code>	Writes the contents of this document to the given output stream.
<code>print(PrintWriter)</code>	Writes the contents of this document to the given output stream.
<code>printExternalDTD(OutputStream)</code>	Writes the contents of this document to the given output stream.
<code>printExternalDTD(OutputStream, String)</code>	Writes the contents of the external DTD to the given output stream.
<code>printExternalDTD(PrintWriter)</code>	Writes the contents of this document to the given output stream.
<code>replaceChild(Node, Node)</code>	Replaces the child node <code>oldChild</code> with <code>newChild</code> in the list of children, and returns the <code>oldChild</code> node.
<code>setEncoding(String)</code>	Sets the character encoding for output.
<code>setLocale(Locale)</code>	Sets the locale for error reporting
<code>setStandalone(String)</code>	Sets the standalone information stored in the <code>&lt;?xml ...?&gt;</code> tag.
<code>setVersion(String)</code>	Sets the version number stored in the <code>&lt;?xml ...?&gt;</code> tag.
<code>validateElementContent(Element)</code>	Validates the content of an element node.

---

---

**Inherited Member Summary**

---

**Fields inherited from class XMLNode**

AMP, ASTERISK, ATTRDECL, cANY, cATTRLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cDREFS, cIGNORE, cIMPLIED, cINCLUDE, cCDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface Node**

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

**Fields inherited from interface oracle.xml.parser.v2.XMLConstants**

AMP, ASTERISK, cANY, cATTRLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cDREFS, cIGNORE, cIMPLIED, cINCLUDE, cCDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Methods inherited from class XMLNode**

appendChild(Node), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Methods inherited from interface Node**

appendChild(Node), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), setNodeValue(String)

---

## Constructor

### XMLDocument()

```
public XMLDocument()  
Creates an empty document.
```

## Methods

### cloneNode(boolean)

```
public org.w3c.dom.Node cloneNode(boolean deep)  
Returns a duplicate of this document node.
```

#### Specified By

org.w3c.dom.Node.cloneNode(boolean) in interface org.w3c.dom.Node

#### Overrides

cloneNode(boolean) in class XMLNode

#### Parameters

deep - If true, recursively clone the subtree under the document; if false, clone only the document itself

#### Returns

The duplicate document node.

### createAttribute(String)

```
public org.w3c.dom.Attr createAttribute(java.lang.String name)  
Creates an Attr of the given name. Note that the Attr instance can then be set on an Element using the setAttribute method.
```

#### Specified By

org.w3c.dom.Document.createAttribute(String) in interface org.w3c.dom.Document

#### Parameters

name - The name of the attribute.



**Returns**

A new `Attr` object.

**Throws**

`org.w3c.dom.DOMException - INVALID_CHARACTER_ERR`: Raised if the specified name contains an invalid character.

**createCDATASection(String)**

```
public org.w3c.dom.CDATASection createCDATASection(java.lang.String data)
```

Creates a `CDATASection` node whose value is the specified string.

**Specified By**

`org.w3c.dom.Document.createCDATASection(String)` in interface `org.w3c.dom.Document`

**Parameters**

`data` - The data for the `CDATASection` contents.

**Returns**

The new `CDATASection` object.

**Throws**

`org.w3c.dom.DOMException` - A `DOMException` could be thrown.

**createComment(String)**

```
public org.w3c.dom.Comment createComment(java.lang.String data)
```

Creates a `Comment` node given the specified string.

**Specified By**

`org.w3c.dom.Document.createComment(String)` in interface `org.w3c.dom.Document`

**Parameters**

`data` - The data for the node.

**Returns**

The new `Comment` object.

**createDocumentFragment()**

```
public org.w3c.dom.DocumentFragment createDocumentFragment()  
Creates an empty DocumentFragment object.
```

**Specified By**

org.w3c.dom.Document.createDocumentFragment() in interface org.w3c.dom.Document

**Returns**

A new DocumentFragment.

**createElement(String)**

```
public org.w3c.dom.Element createElement(java.lang.String tagName)  
Creates an element of the type specified. Note that the instance returned  
implements the Element interface, so attributes can be specified directly on the  
returned object.
```

**Specified By**

org.w3c.dom.Document.createElement(String) in interface org.w3c.dom.Document

**Parameters**

tagName - The name of the element type to instantiate. The name is treated as case-sensitive.

**Returns**

A new Element object.

**Throws**

org.w3c.dom.DOMException - INVALID\_CHARACTER\_ERR: Raised if the specified name contains an invalid character.

**createEntityReference(String)**

```
public org.w3c.dom.EntityReference createEntityReference(java.lang.String name)  
Creates an EntityReference object.
```

**Specified By**

org.w3c.dom.Document.createEntityReference(String) in interface org.w3c.dom.Document

**Parameters**

name - The name of the entity to reference.

**Returns**

The new `EntityReference` object.

**Throws**

`org.w3c.dom.DOMException - INVALID_CHARACTER_ERR`: Raised if the specified name contains an invalid character.

**createProcessingInstruction(String, String)**

```
public org.w3c.dom.ProcessingInstruction  
createProcessingInstruction(java.lang.String target, java.lang.String data)  
Creates a ProcessingInstruction node given the specified name and data  
strings.
```

**Specified By**

`org.w3c.dom.Document.createProcessingInstruction(String, String)` in interface  
`org.w3c.dom.Document`

**Parameters**

target - The target part of the processing instruction.

data - The data for the node.

**Returns**

The new `ProcessingInstruction` object.

**Throws**

`org.w3c.dom.DOMException - INVALID_CHARACTER_ERR`: Raised if an invalid character is specified.

**createTextNode(String)**

```
public org.w3c.dom.Text createTextNode(java.lang.String data)  
Creates a Text node given the specified string.
```

### Specified By

`org.w3c.dom.Document.createTextNode(String)` in interface `org.w3c.dom.Document`

### Parameters

`data` - The data for the node.

### Returns

The new `Text` object.

### `expectedElements(Element)`

```
public java.util.Vector expectedElements(org.w3c.dom.Element e)
```

Returns vector of element names that can be appended to the element.

### Parameters

`e` - Element

### Returns

Vector of names

### `getDoctype()`

```
public org.w3c.dom.DocumentType getDoctype()
```

The Document Type Declaration (DTD) associated with this document. For XML documents without a DTD, this returns `null`. Note that the DOM Level 1 specification does not support editing the DTD.

### Specified By

`org.w3c.dom.Document.getDoctype()` in interface `org.w3c.dom.Document`

### Returns

The associated DTD

### See Also

`org.w3c.dom.DocumentType`

### `getDocumentElement()`

```
public org.w3c.dom.Element getDocumentElement()
```

This is a convenience attribute that allows direct access to the child node that is the root element of the document.

### Specified By

`org.w3c.dom.Document.getDocumentElement()` in interface `org.w3c.dom.Document`

### Returns

The root element

## **getElementsByTagName(String)**

```
public org.w3c.dom.NodeList getElementsByTagName(java.lang.String tagname)
```

Returns a `NodeList` of all the `Elements` with a given tag name in the order in which they would be encountered in a preorder traversal of the `Document` tree.

### Specified By

`org.w3c.dom.Document.getElementsByTagName(String)` in interface `org.w3c.dom.Document`

### Parameters

`tagname` - The name of the tag to match on. The special value "\*" matches all tags.

### Returns

A new `NodeList` object containing all the matched `Elements`.

## **getEncoding()**

```
public final java.lang.String getEncoding()
```

Retrieves the character encoding information.

### Returns

the encoding information stored in the `<?xml ...?>` tag or the user-defined output encoding if it has been more recently set.

## **getImplementation()**

```
public org.w3c.dom.DOMImplementation getImplementation()
```

The `DOMImplementation` object that handles this document. A DOM application may use objects from multiple implementations.

### Specified By

`org.w3c.dom.Document.getImplementation()` in interface `org.w3c.dom.Document`

### Returns

The associated DOM implementation.

### getOwnerDocument()

```
public org.w3c.dom.Document getOwnerDocument()
```

The `Document` object associated with this node. Since this node is a `Document` this is `null`.

### Specified By

`org.w3c.dom.Node.getOwnerDocument()` in interface `org.w3c.dom.Node`

### Overrides

`getOwnerDocument()` in class `XMLNode`

### Returns

`null`

### getStandalone()

```
public final java.lang.String getStandalone()
```

Retrieves the standalone information.

### Returns

the standalone attribute stored in the `<?xml ...?>` tag.

### getVersion()

```
public final java.lang.String getVersion()
```

Retrieves the version information.

### Returns

the version number stored in the `<?xml ...?>` tag.

### print(OutputStream)

```
public void print(java.io.OutputStream out)
```

Writes the contents of this document to the given output stream.

**Overrides**

print(OutputStream) in class XMLNode

**Parameters**

out - OutputStream to write to

**Throws**

IOException - if an error occurs

**print(OutputStream, String)**

```
public void print(java.io.OutputStream out, java.lang.String enc)
```

Writes the contents of this document to the given output stream.

**Overrides**

print(OutputStream, String) in class XMLNode

**Parameters**

out - OutputStream to write to

enc - Encoding to use for the output

**Throws**

IOException - if an invalid encoding was specified or if any other error occurs

**print(PrintWriter)**

```
public void print(java.io.PrintWriter out)
```

Writes the contents of this document to the given output stream.

**Overrides**

print(PrintWriter) in class XMLNode

**Parameters**

out - PrintWriter to write to

### Throws

`IOException` - if an error occurs

### **printExternalDTD(OutputStream)**

```
public void printExternalDTD(java.io.OutputStream out)
```

Writes the contents of this document to the given output stream.

### Parameters

`out` - `OutputStream` to write to

### Throws

`IOException` - if an error occurs

### **printExternalDTD(OutputStream, String)**

```
public void printExternalDTD(java.io.OutputStream out, java.lang.String enc)
```

Writes the contents of the external DTD to the given output stream.

### Parameters

`out` - `OutputStream` to write to

`enc` - Encoding to use for the output

### Throws

`IOException` - if an invalid encoding was specified or if any other error occurs

### **printExternalDTD(PrintWriter)**

```
public void printExternalDTD(java.io.PrintWriter out)
```

Writes the contents of this document to the given output stream.

### Parameters

`out` - `PrintWriter` to write to

### Throws

`IOException` - if an error occurs

### **replaceChild(Node, Node)**

```
public org.w3c.dom.Node replaceChild(org.w3c.dom.Node newChild,
```



`org.w3c.dom.Node oldChild)`

Replaces the child node `oldChild` with `newChild` in the list of children, and returns the `oldChild` node. If the `newChild` is already in the tree, it is first removed. This is an override of the `XMLNode.removeChild` method

### Specified By

`org.w3c.dom.Node.replaceChild(Node, Node)` in interface `org.w3c.dom.Node`

### Overrides

`replaceChild(Node, Node)` in class `XMLNode`

### Parameters

`newChild` - The new node to put in the child list.

`oldChild` - The node being replaced in the list.

### Returns

The node replaced.

### Throws

`org.w3c.dom.DOMException - HIERARCHY_REQUEST_ERR`: Raised if this node is of a type that does not allow children of the type of the `newChild` node.

`WRONG_DOCUMENT_ERR`: Raised if `newChild` was created from a different document than this one. `NOT_FOUND_ERR`: Raised if `oldChild` is not a child of this node.

## setEncoding(String)

`public final void setEncoding(java.lang.String encoding)`

Sets the character encoding for output. Eventually it sets the `ENCODING` stored in the `<?xml ...?>` tag, but not until the document is saved. You should not call this method until the Document has been loaded.

### Parameters

`encoding` - The character encoding to set

## setLocale(Locale)

`public final void setLocale(java.util.Locale locale)`

Sets the locale for error reporting

### Parameters

`locale` - Locale for error reporting.

### setStandalone(String)

```
public final void setStandalone(java.lang.String value)
```

Sets the standalone information stored in the `<?xml ...?>` tag.

### Parameters

`value` - The attribute value ('yes' or 'no').

### setVersion(String)

```
public final void setVersion(java.lang.String version)
```

Sets the version number stored in the `<?xml ...?>` tag.

### Parameters

`version` - The version information to set.

### validateElementContent(Element)

```
public boolean validateElementContent(org.w3c.dom.Element e)
```

Validates the content of a element node.

### Parameters

`e` - Element to be validated

### Returns

True if valid, else false

---

# XMLDocumentFragment

## Syntax

```
public class XMLDocumentFragment extends XMLNode implements  
org.w3c.dom.DocumentFragment, java.io.Serializable
```

```
java.lang.Object  
|  
+--XMLNode  
|  
+--oracle.xml.parser.v2.XMLDocumentFragment
```

## All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.DocumentFragment, org.w3c.dom.Node,  
java.io.Serializable, oracle.xml.parser.v2.XMLConstants
```

## Description

This class implements the DOM DocumentFragment interface.

## See Also

[org.w3c.dom.DocumentFragment](#), [NodeFactory](#), [setNodeFactory\(NodeFactory\)](#)

---

## Member Summary

### Constructors

<code>XMLDocumentFragment()</code>	Creates an empty document fragment
------------------------------------	------------------------------------

### Methods

<code>getParentNode()</code>	Gets the parent of this node
------------------------------	------------------------------

---

---

## Inherited Member Summary

Fields inherited from class `XMLNode`



## Constructor

### XMLDocumentFragment()

```
public XMLDocumentFragment()  
Creates an empty document fragment
```

## Methods

### getParentNode()

```
public org.w3c.dom.Node getParentNode()  
Gets the parent of this node
```

### Specified By

org.w3c.dom.Node.getParentNode() in interface org.w3c.dom.Node

### Overrides

getParentNode() in class XMLNode

### Returns

The parent of this node (always null)

## XMLDocumentHandler

### Syntax

```
public interface XMLDocumentHandler extends org.xml.sax.DocumentHandler
```

### All Superinterfaces

```
org.xml.sax.DocumentHandler
```

### All Known Implementing Classes

```
Package oracle.xml.parser.v2
```

### Description

This interface extends the `org.xml.sax.DocumentHandler` interface. SAX Applications requiring Namespace support must implement this interface and register with the SAX Parser via `Parser.setDocumentHandler()`.

---

### Member Summary

---

#### Methods

<code>cDATASection(char[], int, int)</code>	Receive notification of a CDATA Section.
<code>comment(String)</code>	Receive notification of a comment.
<code>endDoctype()</code>	Receive notification of end of the DTD.
<code>endElement(NSName)</code>	Receive notification of the end of an element.
<code>setDoctype(DTD)</code>	Receive notification of a DTD (Document Type node).
<code>setTextDecl(String, String)</code>	Receive notification of a Text XML Declaration.
<code>setXMLDecl(String, String, String)</code>	Receive notification of a XML Declaration.
<code>startElement(NSName, SAXAttrList)</code>	Receive notification of the beginning of an element.

---

---

### Inherited Member Summary

---

#### Methods inherited from interface DocumentHandler

`characters(char[], int, int)`, `endDocument()`, `endElement(String)`, `ignorableWhitespace(char[], int, int)`, `processingInstruction(String, String)`, `setDocumentLocator(Locator)`, `startDocument()`, `startElement(String, AttributeList)`

---

## Methods

### **cDATASection(char[ ], int, int)**

```
public void cDATASection(char[] ch, int start, int length)
```

Receive notification of a CDATA Section.

The Parser will invoke this method once for each CDATA Section found.

#### **Parameters**

*ch* - The CDATA section characters.

*start* - The start position in the character array.

*length* - The number of characters to use from the character array.

#### **Throws**

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

### **comment(String)**

```
public void comment(java.lang.String data)
```

Receive notification of a comment.

The Parser will invoke this method once for each comment found note that comment may occur before or after the main document element.

#### **Parameters**

*data* - The comment data, or null if none was supplied.

#### **Throws**

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

### **endDoctype()**

```
public void endDoctype()
```

Receive notification of end of the DTD.

### Throws

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

### `endElement(NSName)`

```
public void endElement(NSName elem)
```

Receive notification of the end of an element. The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding `startElement()` event for every `endElement()` event (even when the element is empty).

By implementing this method instead of `org.xml.sax.DocumentHandler.endElement`, SAX Applications can get the Namespace support provided by `NSName`.

### Parameters

`elem` - `NSName` object

### Throws

`org.xml.sax.SAXException` - A `SAXException` could be thrown.

**See Also:** `org.xml.sax.DocumentHandler.endElement(String)`

### `setDoctype(DTD)`

```
public void setDoctype(DTD dtd)
```

Receive notification of a DTD (Document Type node).

The Parser will invoke this method after calling `startDocument` to register the DTD used.

### Parameters

`DTD` - The DTD node

### Throws

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

### `setTextDecl(String, String)`

```
public void setTextDecl(java.lang.String version, java.lang.String encoding)
```



Receive notification of a Text XML Declaration.

The Parser will invoke this method once for each text XML Decl

### Parameters

`version` - The version number (or null, if not specified)

`encoding` - The encoding name

### Throws

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

## **setXMLDecl(String, String, String)**

```
public void setXMLDecl(java.lang.String version, java.lang.String standalone,  
java.lang.String encoding)
```

Receive notification of a XML Declaration.

The Parser will invoke this method once for XML Decl

### Parameters

`version` - The version number

`standalone` - The standalone value (or null, if not specified)

`encoding` - The encoding name (or null, if not specified)

### Throws

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

## **startElement(NSName, SAXAttrList)**

```
public void startElement(NSName elem, SAXAttrList attrlist)
```

Receive notification of the beginning of an element. The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding `endElement()` event for every `startElement()` event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding `endElement()` event.

By implementing this method instead of `org.xml.sax.DocumentHandler.startElement`, SAX Applications can get the Namespace support provided by `NSName` and `SAXAttrList`.

**Parameters**

`elem` - `NSName` object

`attrlist` - `SAXAttrList` for the element

**Throws**

`org.xml.sax.SAXException` - A `SAXException` could be thrown.

**See Also** `org.xml.sax.DocumentHandler.startElement(String, AttributeList)`:

# XMLElement

## Syntax

public class XMLElement extends XMLNode implements org.w3c.dom.Element, java.io.Serializable, NSName, NSResolver

```

java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.XMLElement

```

## All Implemented Interfaces

ava.lang.Cloneable, org.w3c.dom.Element, org.w3c.dom.Node, NSName, NSResolver, java.io.Serializable, oracle.xml.parser.v2.XMLConstants

## Description

This class implements the DOM `Element` interface. Elements are created by the XML parser using the default `NodeFactory` or the user defined `NodeFactory` if registered using `setNodeFactory()` method.

**See Also:** `org.w3c.dom.Element`, `XMLParser`, `NodeFactory`, `setNodeFactory(NodeFactory)`:

---

## Member Summary

---

### Constructors

XMLElement(String)	Creates an element with the given name
XMLElement(String, String, String)	Creates an element with the given name, prefix, and namespace

### Methods

checkNamespace(String, String)	Returns if the element belongs to the namespace specified.
cloneNode(boolean)	Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
getAttribute(String)	Retrieves an attribute value by name.
getAttributeNode(String)	Retrieves an <code>Attr</code> node by name.

---

### Member Summary

---

<code>getAttributes()</code>	A <code>NamedNodeMap</code> containing the attributes of this node (if it is an <code>Element</code> ) or <code>null</code> otherwise.
<code>getChildrenByTagName(String)</code>	Returns a <code>NodeList</code> of all immediate children with a given tag name,
<code>getChildrenByTagName(String, String)</code>	Returns a <code>NodeList</code> of all immediate children with a given tag name and namespace
<code>getElementsByTagName(String)</code>	Returns a <code>NodeList</code> of all descendant elements with a given tag name, in the order in which they would be encountered in a preorder traversal of the <code>Element</code> tree.
<code>getElementsByTagName(String, String)</code>	Returns a <code>NodeList</code> of all descendant elements with a given tag name, and namespace in the order in which they would be encountered in a preorder traversal of the <code>Element</code> tree.
<code>getExpandedName()</code>	Get the fully resolved name for this element.
<code>getLocalName()</code>	Get the local Name for this element.
<code>getNamespace()</code>	Get the resolved Namespace for this element.
<code>getPrefix()</code>	Get the namespace prefix for this element.
<code>getQualifiedName()</code>	Get the qualified name for this element.
<code>getTagName()</code>	Gets the name of the element.
<code>normalize()</code>	Puts all <code>Text</code> nodes in the full depth of the sub-tree underneath this <code>Element</code> into a "normal" form where only markup (e.g., tags, comments, processing instructions, CDATA sections, and entity references) separates <code>Text</code> nodes, i.e., there are no adjacent <code>Text</code> nodes.
<code>removeAttribute(String)</code>	Removes an attribute by name.
<code>removeAttributeNode(Attr)</code>	Removes the specified attribute.
<code>resolveNamespacePrefix(String)</code>	Given a namespace prefix, find the namespace definition in scope in this element.
<code>setAttribute(String, String)</code>	Adds a new attribute.
<code>setAttributeNode(Attr)</code>	Adds a new attribute.

---

---

### Inherited Member Summary

---

Fields inherited from class `XMLNode`

---

**Inherited Member Summary**

---

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cCDATA, cNMTOKE, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Fields inherited from interface Node**

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

**Fields inherited from interface oracle.xml.parser.v2.XMLConstants**

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cCDATA, cNMTOKE, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

**Methods inherited from class XMLNode**

appendChild(Node), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Methods inherited from interface Node**

appendChild(Node), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling( ), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Constructor

### **XMLElement(String)**

```
public XMLElement(java.lang.String tag)
Creates an element with the given name
```

### **XMLElement(String, String, String)**

```
public XMLElement(java.lang.String name, java.lang.String prefix,
java.lang.String namespace)
Creates an element with the given name, prefix, and namespace
```

## Methods

### **checkNamespace(String, String)**

```
public boolean checkNamespace(java.lang.String localname, java.lang.String ns)
Returns if the element belongs to the namespace specified.
```

#### **Parameters**

ns - Expanded namespace string

#### **Returns**

true - if the element belongs to the namespace

### **cloneNode(boolean)**

```
public org.w3c.dom.Node cloneNode(boolean deep)
Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
The duplicate node has no parent (parentNode returns null.). Cloning an
Element copies all attributes and their values, including those generated by the
XML processor to represent defaulted attributes, but this method does not copy any
text it contains unless it is a deep clone, since the text is contained in a child Text
node. Cloning any other type of node simply returns a copy of this node.
```

#### **Specified By**

org.w3c.dom.Node.cloneNode(boolean) in interface org.w3c.dom.Node

#### **Specified By**

org.w3c.dom.Node.cloneNode(boolean) in interface org.w3c.dom.Node

**Overrides**

cloneNode(boolean) in class XMLNode

**Parameters**

deep - If true, recursively clone the subtree under the specified node; if false, clone only the node itself (and its attributes, if it is an Element).

**Returns**

The duplicate node.

**getAttribute(String)**

```
public java.lang.String getAttribute(java.lang.String name)
```

Retrieves an attribute value by name.

**Specified By**

org.w3c.dom.Element.getAttribute(String) in interface org.w3c.dom.Element

**Parameters**

name - The name of the attribute to retrieve.

**Returns**

The Attr value as a string, or the empty string if that attribute does not have a specified or default value.

**getAttributeNode(String)**

```
public org.w3c.dom.Attr getAttributeNode(java.lang.String name)
```

Retrieves an Attr node by name.

**Specified By**

org.w3c.dom.Element.getAttributeNode(String) in interface org.w3c.dom.Element

**Parameters**

name - The name of the attribute to retrieve.

**Returns**

The `Attr` node with the specified attribute name or `null` if there is no such attribute.

**getAttributes()**

```
public org.w3c.dom.NamedNodeMap getAttributes()
```

A `NamedNodeMap` containing the attributes of this node (if it is an `Element`) or `null` otherwise.

**Specified By**

`org.w3c.dom.Node.getAttributes()` in interface `org.w3c.dom.Node`

**Overrides**

`getAttributes()` in class `XMLNode`

**Returns**

The list of attributes of this element

**getChildrenByTagName(String)**

```
public org.w3c.dom.NodeList getChildrenByTagName(java.lang.String name)
```

Returns a `NodeList` of all immediate children with a given tag name,

**Parameters**

`name` - The name of the tag to match on.

**Returns**

A list of matching children

**getChildrenByTagName(String, String)**

```
public org.w3c.dom.NodeList getChildrenByTagName(java.lang.String name,  
java.lang.String ns)
```

Returns a `NodeList` of all immediate children with a given tag name and namespace

**Parameters**

`name` - The name of the tag to match on. (should be local name)

`ns` - The name space



**Returns**

A list of matching children

**getElementsByTagName(String)**

```
public org.w3c.dom.NodeList getElementsByTagName(java.lang.String name)
```

Returns a `NodeList` of all descendant elements with a given tag name, in the order in which they would be encountered in a preorder traversal of the `Element` tree.

**Specified By**

`org.w3c.dom.Element.getElementsByTagName(String)` in interface `org.w3c.dom.Element`

**Parameters**

`name` - The name of the tag to match on. The special value "\*" matches all tags.

**Returns**

A list of matching `Element` nodes.

**getElementsByTagName(String, String)**

```
public org.w3c.dom.NodeList getElementsByTagName(java.lang.String name,
java.lang.String ns)
```

Returns a `NodeList` of all descendant elements with a given tag name, and namespace in the order in which they would be encountered in a preorder traversal of the `Element` tree.

**Parameters**

`name` - The name of the tag to match on. The special value "\*" matches all tags. (should be local name)

`ns` - The namespace of the elements

**Returns**

A list of matching `Element` nodes.

**getExpandedName()**

```
public java.lang.String getExpandedName()
```

Get the fully resolved name for this element.

**Specified By**

getExpandedName() in interface NSName

**Returns**

the fully resolved name

**getLocalName()**

public java.lang.String getLocalName()  
Get the local Name for this element.

**Specified By**

getLocalName() in interface NSName

**Returns**

the local Name

**getNamespace()**

public java.lang.String getNamespace()  
Get the resolved Namespace for this element.

**Specified By**

getNamespace() in interface NSName

**Returns**

the resolved Namespace

**getPrefix()**

public java.lang.String getPrefix()  
Get the namespace prefix for this element.

**Specified By**

getPrefix() in interface NSName

**Returns**

the namespace prefix

## getQualifiedName()

```
public java.lang.String getQualifiedName()  
Get the qualified name for this element.
```

### Specified By

getQualifiedName() in interface NSName

### Returns

the qualified name

## getTagName()

```
public java.lang.String getTagName()  
Gets the name of the element. For example, in <elementExample id="demo"> ...  
</elementExample>, tagName has the value "elementExample". Note that this  
is case-preserving in XML, as are all of the operations of the DOM. The HTML  
DOM returns the tagName of an HTML element in the canonical uppercase form,  
regardless of the case in the source HTML document.
```

### Specified By

org.w3c.dom.Element.getTagName() in interface org.w3c.dom.Element

### Returns

The element name

## normalize()

```
public void normalize()  
Puts all Text nodes in the full depth of the sub-tree underneath this Element into  
a "normal" form where only markup (e.g., tags, comments, processing instructions,  
CDATA sections, and entity references) separates Text nodes, i.e., there are no  
adjacent Text nodes. This can be used to ensure that the DOM view of a document  
is the same as if it were saved and re-loaded, and is useful when operations (such as  
XPath lookups) that depend on a particular document tree structure are to be  
used.
```

### Specified By

org.w3c.dom.Element.normalize() in interface org.w3c.dom.Element

## removeAttribute(String)

```
public void removeAttribute(java.lang.String name)
```

Removes an attribute by name. If the removed attribute has a default value it is immediately replaced.

### Specified By

`org.w3c.dom.Element.removeAttribute(String)` in interface `org.w3c.dom.Element`

### Parameters

`name` - The name of the attribute to remove.

### Throws

`org.w3c.dom.DOMException - NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly.

## removeAttributeNode(Attr)

```
public org.w3c.dom.Attr removeAttributeNode(org.w3c.dom.Attr oldAttr)
```

Removes the specified attribute.

### Specified By

`org.w3c.dom.Element.removeAttributeNode(Attr)` in interface `org.w3c.dom.Element`

### Parameters

`oldAttr` - The `Attr` node to remove from the attribute list. If the removed `Attr` has a default value it is immediately replaced.

### Returns

The `Attr` node that was removed.

### Throws

`org.w3c.dom.DOMException - NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly. `NOT_FOUND_ERR`: Raised if `oldAttr` is not an attribute of the element.

## resolveNamespacePrefix(String)

```
public java.lang.String resolveNamespacePrefix(java.lang.String prefix)
```

Given a namespace prefix, find the namespace definition in scope in this element.

**Specified By**

resolveNamespacePrefix(String) in interface NSResolver

**Parameters**

`prefix` - Namespace prefix to be resolved

**Returns**

the resolved Namespace (null, if prefix could not be resolved)

**setAttribute(String, String)**

```
public void setAttribute(java.lang.String name, java.lang.String value)
```

Adds a new attribute. If an attribute with that name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string, it is not parsed as it is being set. So any markup (such as syntax to be recognized as an entity reference) is treated as literal text, and needs to be appropriately escaped by the implementation when it is written out. In order to assign an attribute value that contains entity references, the user must create an `Attr` node plus any `Text` and `EntityReference` nodes, build the appropriate subtree, and use `setAttributeNode` to assign it as the value of an attribute.

**Specified By**

org.w3c.dom.Element.setAttribute(String, String) in interface org.w3c.dom.Element

**Parameters**

`name` - The name of the attribute to create or alter.

`value` - Value to set in string form.

**Throws**

org.w3c.dom.DOMException - `INVALID_CHARACTER_ERR`: Raised if the specified name contains an invalid character. `NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly.

**setAttributeNode(Attr)**

```
public org.w3c.dom.Attr setAttributeNode(org.w3c.dom.Attr newAttr)
```

Adds a new attribute. If an attribute with that name is already present in the element, it is replaced by the new one.

### Specified By

`org.w3c.dom.Element.setAttributeNode(Attr)` in interface `org.w3c.dom.Element`

### Parameters

`newAttr` - The `Attr` node to add to the attribute list.

### Returns

If the `newAttr` attribute replaces an existing attribute with the same name, the previously existing `Attr` node is returned, otherwise `null` is returned.

### Throws

`org.w3c.dom.DOMException` - `WRONG_DOCUMENT_ERR`: Raised if `newAttr` was created from a different document than the one that created the element.  
`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is `readonly`.  
`INUSE_ATTRIBUTE_ERR`: Raised if `newAttr` is already an attribute of another `Element` object. The DOM user must explicitly clone `Attr` nodes to re-use them in other elements.

# XMLEntityReference

## Syntax

```
public class XMLEntityReference extends XMLNode implements
org.w3c.dom.EntityReference, oracle.xml.parser.v2.XMLConstants,
java.lang.Cloneable, java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.XMLEntityReference
```

## All Implemented Interfaces

```
java.lang.Cloneable, org.w3c.dom.EntityReference, org.w3c.dom.Node,
java.io.Serializable, oracle.xml.parser.v2.XMLConstants
```

## Description

---

### Member Summary

---

#### Constructors

```
XMLEntityReference(String)
```

---



---

### Inherited Member Summary

---

#### Fields inherited from class XMLNode

AMP, ASTERISK, ATTRDECL, cANY, cATTRLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

#### Fields inherited from interface Node

---

### Inherited Member Summary

---

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

#### Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

#### Methods inherited from class XMLNode

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), setNodeValue(String), transformNode(XSLStylesheet), valueOf(String, NSResolver)

#### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

#### Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

---

## Constructor

### XMLEntityReference(String)

```
public XMLEntityReference(java.lang.String tag)
```



# XMLNode

## Syntax

public class XMLNode extends java.lang.Object implements org.w3c.dom.Node, oracle.xml.parser.v2.XMLConstants, java.lang.Cloneable, java.io.Serializable

```
java.lang.Object
|
+--oracle.xml.parser.v2.XMLNode
```

## Direct Known Subclasses

AttrDecl, oracle.xml.parser.v2.CharData, DTD, ElementDecl, XMLAttr, XMLDocument, XMLDocumentFragment, XMLElement, XMLEntityReference

## All Implemented Interfaces

java.lang.Cloneable, org.w3c.dom.Node, java.io.Serializable, oracle.xml.parser.v2.XMLConstants

## Description

Implements the DOM `Node` interface and serves as the primary datatype for the entire Document Object Model. It represents a single node in the document tree.

The attributes `nodeName`, `nodeValue` and `attributes` are included as a mechanism to get at node information without casting down to the specific derived instance. In cases where there is no obvious mapping of these attributes for a specific `nodeType` (e.g., `nodeValue` for an `Element` or `attributes` for a `Comment`), this returns `null`. Note that the derived classes may contain additional and more convenient mechanisms to get and set the relevant information.

---

## Member Summary

---

### Fields

ATTRDECL	A attribute declaration node
ELEMENTDECL	An element declaration node.

### Methods

appendChild(Node)	Adds the node <code>newChild</code> to the end of the list of children of this node.
cloneNode(boolean)	Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.

---

**Member Summary**

---

<code>getAttributes()</code>	Gets a <code>NamedNodeMap</code> containing the attributes of this node (if it is an <code>Element</code> ) or <code>null</code> otherwise.
<code>getChildNodes()</code>	Gets a <code>NodeList</code> that contains all children of this node.
<code>getFirstChild()</code>	Gets the first child of this node.
<code>getLastChild()</code>	Gets the last child of this node.
<code>getNextSibling()</code>	Gets The node immediately following this node.
<code>getNodeName()</code>	Gets the name of this node, depending on its type
<code>getNodeType()</code>	Gets a code representing the type of the underlying object
<code>getNodeValue()</code>	Gets the value of this node, depending on its type
<code>getOwnerDocument()</code>	Gets the <code>Document</code> object associated with this node.
<code>getParentNode()</code>	Gets the parent of this node.
<code>getPreviousSibling()</code>	Gets the node immediately preceding this node.
<code>hasChildNodes()</code>	This is a convenience method to allow easy determination of whether a node has any children.
<code>insertBefore(Node, Node)</code>	Inserts the node <code>newChild</code> before the existing child node <code>refChild</code> .
<code>print(OutputStream)</code>	Writes the contents of this node to the given output stream.
<code>print(OutputStream, String)</code>	Writes the contents of this node to the given output stream.
<code>print(PrintWriter)</code>	Writes the contents of this node using the given print writer.
<code>removeChild(Node)</code>	Removes the child node indicated by <code>oldChild</code> from the list of children, and returns it.
<code>replaceChild(Node, Node)</code>	Replaces the child node <code>oldChild</code> with <code>newChild</code> in the list of children, and returns the <code>oldChild</code> node.
<code>selectNodes(String, NSResolver)</code>	Selects nodes from the tree which match the given pattern
<code>selectSingleNode(String, NSResolver)</code>	Selects the first node from the tree that matches the given pattern
<code>setNodeValue(String)</code>	Sets the value of this node, depending on its type
<code>transformNode(XSLStylesheet)</code>	Transforms a node in the tree using the given stylesheet
<code>valueOf(String, NSResolver)</code>	Selects the value of the first node from the tree that matches the given pattern

---

---

## Inherited Member Summary

---

Fields inherited from interface Node

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNM\_TOKEN, cNM\_TOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

## Fields

### ATTRDECL

```
public static final short ATTRDECL
A attribute declaration node
```

### ELEMENTDECL

```
public static final short ELEMENTDECL
An element declaration node.
```

## Methods

### appendChild(Node)

```
public org.w3c.dom.Node appendChild(org.w3c.dom.Node newChild)
Adds the node newChild to the end of the list of children of this node. If the newChild is already in the tree, it is first removed.
```

### Specified By

org.w3c.dom.Node.appendChild(Node) in interface org.w3c.dom.Node

**Parameters**

`newChild` - The node to add. If it is a `DocumentFragment` object, the entire contents of the document fragment are moved into the child list of this node

**Returns**

The node added.

**Throws**

`org.w3c.dom.DOMException` - `HIERARCHY_REQUEST_ERR`: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to append is one of this node's ancestors. `WRONG_DOCUMENT_ERR`: Raised if `newChild` was created from a different document than the one that created this node. `NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly.

**cloneNode(boolean)**

```
public org.w3c.dom.Node cloneNode(boolean deep)
```

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (`parentNode` returns `null`). Cloning an `Element` copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child `Text` node. Cloning any other type of node simply returns a copy of this node.

**Specified By**

`org.w3c.dom.Node.cloneNode(boolean)` in interface `org.w3c.dom.Node`

**Parameters**

`deep` - If `true`, recursively clone the subtree under the specified node; if `false`, clone only the node itself (and its attributes, if it is an `Element`).

**Returns**

The duplicate node.

**getAttributes()**

```
public org.w3c.dom.NamedNodeMap getAttributes()
```

Gets a `NamedNodeMap` containing the attributes of this node (if it is an `Element`) or `null` otherwise.

**Specified By**

`org.w3c.dom.Node.getAttributes()` in interface `org.w3c.dom.Node`

**Returns**

the attributes of this node

**getChildNodes()**

```
public org.w3c.dom.NodeList getChildNodes()
```

Gets a `NodeList` that contains all children of this node. If there are no children, this is a `NodeList` containing no nodes. The content of the returned `NodeList` is "live" in the sense that, for instance, changes to the children of the node object that it was created from are immediately reflected in the nodes returned by the `NodeList` accessors; it is not a static snapshot of the content of the node. This is true for every `NodeList`, including the ones returned by the `getElementsByTagName` method.

**Specified By**

`org.w3c.dom.Node.getChildNodes()` in interface `org.w3c.dom.Node`

**Returns**

The children of this node

**getFirstChild()**

```
public org.w3c.dom.Node getFirstChild()
```

Gets the first child of this node. If there is no such node, this returns `null`.

**Specified By**

`org.w3c.dom.Node.getFirstChild()` in interface `org.w3c.dom.Node`

**Returns**

The first child of this node

**getLastChild()**

```
public org.w3c.dom.Node getLastChild()
```

Gets the last child of this node. If there is no such node, this returns `null`.

**Specified By**

`org.w3c.dom.Node.getLastChild()` in interface `org.w3c.dom.Node`

**Returns**

The last child of this node

**getNextSibling()**

```
public org.w3c.dom.Node getNextSibling()
```

Gets The node immediately following this node. If there is no such node, this returns null.

**Specified By**

org.w3c.dom.Node.getNextSibling() in interface org.w3c.dom.Node

**Returns**

the next node

**getNodeName()**

```
public java.lang.String getNodeName()
```

Gets the name of this node, depending on its type

**Specified By**

org.w3c.dom.Node.getNodeName() in interface org.w3c.dom.Node

**Returns**

Name of this node

**getNodeType()**

```
public short getNodeType()
```

Gets a code representing the type of the underlying object

**Specified By**

org.w3c.dom.Node.getNodeType() in interface org.w3c.dom.Node

**Returns**

type of the node

**getNodeValue()**

```
public java.lang.String getNodeValue()
```

Gets the value of this node, depending on its type

**Specified By**

`org.w3c.dom.Node.getNodeValue()` in interface `org.w3c.dom.Node`

**Returns**

Value of this node

**Throws**

`org.w3c.dom.DOMException - NO_MODIFICATION_ALLOWED_ERR`: Raised when the node is readonly. `DOMSTRING_SIZE_ERR`: Raised when it would return more characters than fit in a `DOMString` variable on the implementation platform.

**getOwnerDocument()**

```
public org.w3c.dom.Document getOwnerDocument()
```

Gets the `Document` object associated with this node. This is also the `Document` object used to create new nodes. When this node is a `Document` this is `null`.

**Specified By**

`org.w3c.dom.Node.getOwnerDocument()` in interface `org.w3c.dom.Node`

**Returns**

The document associated with this node

**getParentNode()**

```
public org.w3c.dom.Node getParentNode()
```

Gets the parent of this node. All nodes, except `Document`, `DocumentFragment`, and `Attr` may have a parent. However, if a node has just been created and not yet added to the tree, or if it has been removed from the tree, this is `null`.

**Specified By**

`org.w3c.dom.Node.getParentNode()` in interface `org.w3c.dom.Node`

**Returns**

The parent of this node

## getPreviousSibling()

```
public org.w3c.dom.Node getPreviousSibling()
```

Gets the node immediately preceding this node. If there is no such node, this returns `null`.

### Specified By

`org.w3c.dom.Node.getPreviousSibling()` in interface `org.w3c.dom.Node`

### Returns

the previous node

## hasChildNodes()

```
public boolean hasChildNodes()
```

This is a convenience method to allow easy determination of whether a node has any children.

### Specified By

`org.w3c.dom.Node.hasChildNodes()` in interface `org.w3c.dom.Node`

### Returns

`true` if the node has any children, `false` if the node has no children.

## insertBefore(Node, Node)

```
public org.w3c.dom.Node insertBefore(org.w3c.dom.Node newChild,  
org.w3c.dom.Node refChild)
```

Inserts the node `newChild` before the existing child node `refChild`. If `refChild` is `null`, insert `newChild` at the end of the list of children. If `newChild` is a `DocumentFragment` object, all of its children are inserted, in the same order, before `refChild`. If the `newChild` is already in the tree, it is first removed.

### Specified By

`org.w3c.dom.Node.insertBefore(Node, Node)` in interface `org.w3c.dom.Node`

### Parameters

`newChild` - The node to insert.

`refChild` - The reference node, i.e., the node before which the new node must be inserted.



**Returns**

The node being inserted.

**Throws**

`org.w3c.dom.DOMException - HIERARCHY_REQUEST_ERR`: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to insert is one of this node's ancestors. `WRONG_DOCUMENT_ERR`: Raised if `newChild` was created from a different document than the one that created this node. `NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly. `NOT_FOUND_ERR`: Raised if `refChild` is not a child of this node.

**print(OutputStream)**

```
public void print(java.io.OutputStream out)
```

Writes the contents of this node to the given output stream.

**Parameters**

`out` - `OutputStream` to write to

**Throws**

`IOException` - if an error occurs

**print(OutputStream, String)**

```
public void print(java.io.OutputStream out, java.lang.String enc)
```

Writes the contents of this node to the given output stream.

**Parameters**

`out` - `OutputStream` to write to

`enc` - Encoding to use for the output

**Throws**

`IOException` - if an invalid encoding was specified or if any other error occurs

**print(PrintWriter)**

```
public void print(java.io.PrintWriter out)
```

Writes the contents of this node using the given print writer.

**Parameters**

`out` - `PrintWriter` to use

**Throws**

`IOException` - if an error occurs

**removeChild(Node)**

```
public org.w3c.dom.Node removeChild(org.w3c.dom.Node oldChild)
```

Removes the child node indicated by `oldChild` from the list of children, and returns it.

**Specified By**

`org.w3c.dom.Node.removeChild(Node)` in interface `org.w3c.dom.Node`

**Parameters**

`oldChild` - The node being removed.

**Returns**

The node removed.

**Throws**

`org.w3c.dom.DOMException` - `NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly. `NOT_FOUND_ERR`: Raised if `oldChild` is not a child of this node.

**replaceChild(Node, Node)**

```
public org.w3c.dom.Node replaceChild(org.w3c.dom.Node newChild,  
org.w3c.dom.Node oldChild)
```

Replaces the child node `oldChild` with `newChild` in the list of children, and returns the `oldChild` node. If the `newChild` is already in the tree, it is first removed.

**Specified By**

`org.w3c.dom.Node.replaceChild(Node, Node)` in interface `org.w3c.dom.Node`

**Parameters**

`newChild` - The new node to put in the child list.

`oldChild` - The node being replaced in the list.

### Returns

The node replaced.

### Throws

`org.w3c.dom.DOMException - HIERARCHY_REQUEST_ERR`: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to put in is one of this node's ancestors. `WRONG_DOCUMENT_ERR`: Raised if `newChild` was created from a different document than the one that created this node. `NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly. `NOT_FOUND_ERR`: Raised if `oldChild` is not a child of this node.

## **selectNodes(String, NSResolver)**

```
public org.w3c.dom.NodeList selectNodes(java.lang.String pattern, NSResolver nsr)
```

Selects nodes from the tree which match the given pattern

### Parameters

`pattern` - XSL pattern to match

`nsr` - NSResolver to resolve any prefixes that occur in given pattern

### Returns

a list of matching nodes

### Throws

`XSLException` - Raised if there is an error while doing the match

## **selectSingleNode(String, NSResolver)**

```
public org.w3c.dom.Node selectSingleNode(java.lang.String pattern, NSResolver nsr)
```

Selects the first node from the tree that matches the given pattern

### Parameters

`pattern` - XSL pattern to match

`nsr` - NSResolver to resolve any prefixes that occur in given pattern

**Returns**

matching node

**Throws**

XSLException - Raised if there is an error while doing the match

**setNodeValue(String)**

```
public void setNodeValue(java.lang.String nodeValue)
```

Sets the value of this node, depending on its type

**Specified By**

org.w3c.dom.Node.setNodeValue(String) in interface org.w3c.dom.Node

**Throws**

org.w3c.dom.DOMException - NO\_MODIFICATION\_ALLOWED\_ERR: Raised when the node is readonly. DOMSTRING\_SIZE\_ERR: Raised when it would return more characters than fit in a DOMString variable on the implementation platform.

**transformNode(XSLStylesheet)**

```
public org.w3c.dom.DocumentFragment transformNode(XSLStylesheet xsl)
```

Transforms a node in the tree using the given stylesheet

**Parameters**

xsl - XSLStylesheet to be used for transformation

**Returns**

a document fragment

**Throws**

XSLException - Raised if there is an error while doing the XSL transformation.

**valueOf(String, NSResolver)**

```
public java.lang.String valueOf(java.lang.String pattern, NSResolver nsr)
```

Selects the value of the first node from the tree that matches the given pattern

**Parameters**

pattern - XSL pattern to match

`nsr` - NSResolver to resolve any prefixes that occur in given pattern

**Returns**

value of the matching node

**Throws**

XSLException - Raised if there is an error while doing the match

## XMLParseException

### Syntax

```
public class XMLParseException extends org.xml.sax.SAXParseException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--org.xml.sax.SAXException
         |
         +--org.xml.sax.SAXParseException
            |
            +--oracle.xml.parser.v2.XMLParseException
```

### All Implemented Interfaces

```
java.io.Serializable
```

### Description

Indicates that a parsing exception occurred while processing an XML document

---

### Member Summary

---

#### Fields

ERROR	Code for non-fatal error
FATAL_ERROR	Code for fatal error
WARNING	Code for warning

#### Constructors

```
XMLParseException(String, String,  
String, int, int, int)
```

#### Methods

getColumnNumber(int)	Get the column number of error at specified index
getException(int)	Get the exception (if exists) that occurred in error at specified index
getLineNumber(int)	Get the line number of error at specified index

---

**Member Summary**

---

<code>getMessage(int)</code>	Get the error message at specified index
<code>getMessageType(int)</code>	Get the type of the error message at specified index
<code>getNumMessages()</code>	Return the total number of errors/warnings found during parsing
<code>getPublicId(int)</code>	Get the public ID of input when error at specified index occurred
<code>getSystemId(int)</code>	Get the system ID of input when error at specified index occurred

---



---

**Inherited Member Summary**

---

Methods inherited from interface `SAXParseException`

`getColumnNumber()`, `getLineNumber()`, `getPublicId()`, `getSystemId()`

Methods inherited from interface `SAXException`

`getException()`, `getMessage()`, `toString()`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace`, `getLocalizedMessage`, `printStackTrace`, `printStackTrace`, `printStackTrace`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

---

**Fields****ERROR**

```
public static final int ERROR
Code for non-fatal error
```

**FATAL\_ERROR**

```
public static final int FATAL_ERROR
Code for fatal error
```

**WARNING**

```
public static final int WARNING
Code for warning
```

## Constructor

### XMLParseException(String, String, String, int, int, int)

```
public XMLParseException(java.lang.String msg, java.lang.String pubId,  
java.lang.String sysId, int line, int col, int type)
```

## Methods

### getColumnNumber(int)

```
public int getColumnNumber(int i)  
Get the column number of error at specified index
```

#### Returns

The column number

### getException(int)

```
public java.lang.Exception getException(int i)  
Get the exception (if exists) that occurred in error at specified index
```

#### Returns

The exception

### getLineNumber(int)

```
public int getLineNumber(int i)  
Get the line number of error at specified index
```

#### Returns

The line number

### getMessage(int)

```
public java.lang.String getMessage(int i)  
Get the error message at specified index
```

#### Returns

The error message



**getMessageType(int)**

```
public int getMessageType(int i)
```

Get the type of the error message at specified index

**Returns**

The error message type

**getNumMessages()**

```
public int getNumMessages()
```

Return the total number of errors/warnings found during parsing

**Returns**

The number of errors/warnings

**getPublicId(int)**

```
public java.lang.String getPublicId(int i)
```

Get the public ID of input when error at specified index occurred

**Returns**

The public ID

**getSystemId(int)**

```
public java.lang.String getSystemId(int i)
```

Get the system ID of input when error at specified index occurred

**Returns**

The system ID

## XMLParser

### Syntax

```
public abstract class XMLParser extends java.lang.Object implements  
oracle.xml.parser.v2.XMLConstants
```

```
java.lang.Object  
|  
+--oracle.xml.parser.v2.XMLParser
```

### Direct Known Subclasses

DOMParser, SAXParser

### All Implemented Interfaces

oracle.xml.parser.v2.XMLConstants

### Description

This class serves as a base class for the `DOMParser` and `SAXParser` classes. It contains methods to parse eXtensible Markup Language (XML) 1.0 documents according to the World Wide Web Consortium (W3C) recommendation. This class can not be instantiated (applications may use the DOM or SAX parser depending on their requirements).

---

### Member Summary

---

#### Methods

<code>getReleaseVersion()</code>	Returns the release version of the Oracle XML Parser
<code>getValidationMode()</code>	Returns the validation mode
<code>parse(InputSource)</code>	Parses the XML from given input source
<code>parse(InputStream)</code>	Parses the XML from given input stream.
<code>parse(Reader)</code>	Parses the XML from given input stream.
<code>parse(String)</code>	Parses the XML from the URL indicated
<code>parse(URL)</code>	Parses the XML document pointed to by the given URL and creates the corresponding XML document hierarchy.
<code>setBaseURL(URL)</code>	Set the base URL for loading external entities and DTDs.
<code>setDoctype(DTD)</code>	Set the DTD

---

## Member Summary

---

<code>setLocale(Locale)</code>	Applications can use this to set the locale for error reporting.
<code>setPreserveWhitespace(boolean)</code>	Set the white space preserving mode
<code>setValidationMode(boolean)</code>	Set the validation mode

---



---

## Inherited Member Summary

---

Fields inherited from interface `oracle.xml.parser.v2.XMLConstants`

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

## Methods

### `getReleaseVersion()`

```
public static java.lang.String getReleaseVersion()
Returns the release version of the Oracle XML Parser
```

#### Returns

the release version string

### `getValidationMode()`

```
public boolean getValidationMode()
Returns the validation mode
```

#### Returns

`true` if the XML parser is validating `false` if not

## parse(InputSource)

```
public final void parse(org.xml.sax.InputSource in)
```

Parses the XML from given input source

### Parameters

`in` - the `org.xml.sax.InputSource` to parse

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

## parse(InputStream)

```
public final void parse(java.io.InputStream in)
```

Parses the XML from given input stream. The base URL should be set for resolving external entities and DTD.

### Parameters

`in` - the `InputStream` containing XML data to parse.

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

**See Also:** `setBaseURL(URL)`:

## parse(Reader)

```
public final void parse(java.io.Reader r)
```

Parses the XML from given input stream. The base URL should be set for resolving external entities and DTD.

### Parameters

`r` - the `Reader` containing XML data to parse.

**Throws**

XMLParseException - if syntax or other error encountered.

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

IOException - IO Error.

**See Also**

setBaseURL(URL)

**parse(String)**

```
public final void parse(java.lang.String in)
```

Parses the XML from the URL indicated

**Parameters**

in - the String containing the URL to parse from

**Throws**

XMLParseException - if syntax or other error encountered.

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

IOException - IO Error.

**parse(URL)**

```
public final void parse(java.net.URL url)
```

Parses the XML document pointed to by the given URL and creates the corresponding XML document hierarchy.

**Parameters**

url - the url points to the XML document to parse.

**Throws**

XMLParseException - if syntax or other error encountered.

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

IOException - IO Error.

### setBaseUrl(URL)

```
public void setBaseUrl(java.net.URL url)
```

Set the base URL for loading external entities and DTDs. This method should be called if the `parse(InputStream)` is used to parse the XML Document

#### Parameters

`url` - The base URL

### setDoctype(DTD)

```
public void setDoctype(DTD dtd)
```

Set the DTD

#### Parameters

`dtd` - DTD to set and used while parsing

### setLocale(Locale)

```
public void setLocale(java.util.Locale locale)
```

Applications can use this to set the locale for error reporting.

#### Parameters

`locale` - Locale to set

#### Throws

`org.xml.sax.SAXException` - A `SAXException` could be thrown.

#### See Also

`org.xml.sax.Parser.setLocale(Locale)`

### setPreserveWhitespace(boolean)

```
public void setPreserveWhitespace(boolean flag)
```

Set the white space preserving mode

#### Parameters

`flag` - preserving mode

**setValidationMode(boolean)**

```
public void setValidationMode(boolean yes)
```

Set the validation mode

**Parameters**

*yes* - determines whether the XML parser should be validating

## XMLPI

### Syntax

public class XMLPI extends oracle.xml.parser.v2.CharData implements org.w3c.dom.ProcessingInstruction, java.io.Serializable

```
java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.CharData
      |
      +--oracle.xml.parser.v2.XMLPI
```

### All Implemented Interfaces

org.w3c.dom.CharacterData, java.lang.Cloneable, org.w3c.dom.Node, org.w3c.dom.ProcessingInstruction, java.io.Serializable, oracle.xml.parser.v2.XMLConstants

### Description

This class implements the DOM Processing Instruction interface.

### See Also

org.w3c.dom.ProcessingInstruction, NodeFactory, setNodeFactory(NodeFactory)

---

### Member Summary

---

#### Constructors

XMLPI(String, String)                      Creates a new ProcessingInstruction node with the given target and the data.

#### Methods

getTarget()                                Returns the target of this PI.

---

---

### Inherited Member Summary

---

Fields inherited from class XMLNode



---

## Inherited Member Summary

---

AMP, ASTERISK, ATTRDECL, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEIN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, ELEMENTDECL, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Fields inherited from interface Node

ATTRIBUTE\_NODE, CDATA\_SECTION\_NODE, COMMENT\_NODE, DOCUMENT\_FRAGMENT\_NODE, DOCUMENT\_NODE, DOCUMENT\_TYPE\_NODE, ELEMENT\_NODE, ENTITY\_NODE, ENTITY\_REFERENCE\_NODE, NOTATION\_NODE, PROCESSING\_INSTRUCTION\_NODE, TEXT\_NODE

### Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEIN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

### Methods inherited from class oracle.xml.parser.v2.CharData

appendData, deleteData, getData, getLength, insertData, replaceData, setData, setNodeValue, substringData

### Methods inherited from class XMLNode

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), print(OutputStream), print(OutputStream, String), print(PrintWriter), removeChild(Node), replaceChild(Node, Node), selectNodes(String, NSResolver), selectSingleNode(String, NSResolver), transformNode(XSLStylesheet), valueOf(String, NSResolver)

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface ProcessingInstruction

getData(), setData(String)

### Methods inherited from interface Node

---

## Inherited Member Summary

---

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getNextSibling(), getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode( ), getPreviousSibling(), hasChildNodes(), insertBefore(Node, Node), removeChild(Node), replaceChild(Node, Node), setNodeValue(String)

Methods inherited from interface CharacterData

appendData(String), deleteData(int, int), getLength(), insertData(int, String), replaceData(int, int, String), substringData(int, int)

---

## Constructor

### XMLPI(String, String)

```
public XMLPI(java.lang.String target, java.lang.String data)
```

Creates a new ProcessingInstruction node with the given target and the data.

#### Parameters

target - The target of this PI

data - The content of this PI

## Methods

### getTarget()

```
public java.lang.String getTarget()
```

Returns the target of this PI. XML defines this as the first token following markup that begins the processing instruction.

#### Specified By

org.w3c.dom.ProcessingInstruction.getTarget() in interface  
org.w3c.dom.ProcessingInstruction

#### Returns

The target of the PI.

# XMLText

## Syntax

```
public class XMLText extends oracle.xml.parser.v2.CharData implements
org.w3c.dom.Text, java.io.Serializable
```

```
java.lang.Object
|
+--XMLNode
   |
   +--oracle.xml.parser.v2.CharData
      |
      +--oracle.xml.parser.v2.XMLText
```

## Direct Known Subclasses:

XMLCDATA

## All Implemented Interfaces

org.w3c.dom.CharacterData, java.lang.Cloneable, org.w3c.dom.Node, java.io.Serializable, org.w3c.dom.Text, oracle.xml.parser.v2.XMLConstants

## Description

This class implements the DOM Text interface.

## See Also

org.w3c.dom.Text, NodeFactory, setNodeFactory(NodeFactory)

---

## Member Summary

---

### Constructors

XMLText(String)

### Methods

getNodeValue()

splitText(int)

Breaks Text node into two Text nodes at specified offset, so they are both siblings, and the node only contains content up to the offset.

---



---

## Inherited Member Summary

---

`appendData(String)`, `deleteData(int, int)`, `getData()`, `getLength()`, `insertData(int, String)`, `replaceData(int, int, String)`, `setData(String)`, `substringData(int, int)`

Methods inherited from interface `Node`

`appendChild(Node)`, `cloneNode(boolean)`, `getAttributes()`, `getChildNodes()`, `getFirstChild()`, `getLastChild()`, `getNextSibling()`, `getNodeName()`, `getNodeType()`, `getOwnerDocument()`, `getParentNode()`, `getPreviousSibling()`, `hasChildNodes()`, `insertBefore(Node, Node)`, `removeChild(Node)`, `replaceChild(Node, Node)`, `setNodeValue(String)`

---

## Constructor

### XMLText(String)

```
public XMLText(java.lang.String text)
```

## Methods

### getNodeValue()

```
public java.lang.String getNodeValue()
```

### Specified By

`org.w3c.dom.Node.getNodeValue()` in interface `org.w3c.dom.Node`

### Overrides

`getNodeValue()` in class `XMLNode`

### splitText(int)

```
public org.w3c.dom.Text splitText(int offset)
```

Breaks `Text` node into two `Text` nodes at specified offset, so they are both siblings, and the node only contains content up to the offset. New node inserted as next sibling contains all content at and after the offset point.

### Specified By

`org.w3c.dom.Text.splitText(int)` in interface `org.w3c.dom.Text`

### Parameters

`offset` - Offset at which to split, starting from 0

### **Returns**

New `Text` node

### **Throws**

`org.w3c.dom.DOMException - INDEX_SIZE_ERR`: Raised if specified offset is negative or greater than number of characters in `data`.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is readonly.

# XMLToken

## Syntax

```
public interface XMLToken
```

## Description

Basic interface for XMLToken

All XMLParser applications with Tokenizer feature must implement this interface. The interface has to be registered using XMLParser method `setTokenHandler(XMLToken handler)`.

If XMLtoken handler != null then for each registered and found token the parser calls the XMLToken call-back method `token(int token, String value)`. During tokenizing the parser doesn't validate the document and doesn't include/read internal/external entities. If XMLtoken handler == null then the parser parses as usual.

A request for XML token is registered (on/off) using XMLParser method `setToken(int token, boolean set)`. The requests could be registered during the parsing (from inside the call-back method) as well.

The XML tokens are defined as public constants in XMLToken interface. They correspond to the XML syntax variables from W3C XML Syntax Specification.

---

## Member Summary

---

### Fields

AttListDecl	AttListDecl ::= '<' '!' 'ATTLIST' S Name AttDef* S? '>'
AttName	AttName ::= Name
Attribute	Attribute ::= AttName Eq AttValue
AttValue	AttValue ::= '"' ([^<&"]   Reference)* '"'
CDSect	CDSect ::= CDStart CData CDEnd
CharData	CharData ::= [^<&]* - ([^<&]* ''])>' [^<&]*

---

**Member Summary**

---

Comment	Comment ::= '<' ! '-' ((Char - '-')   ('-' (Char - '-')))* '->'
DTDName	DTDName ::= name
ElemDeclName	ElemDeclName ::= name
elementdecl	elementdecl ::= '<' !ELEMENT' S ElemDeclName S contentspec S? '>'
EmptyElemTag	EmptyElemTag ::= '<' STagName (S Attribute)* S? '/' '>'
EntityDecl	EntityDecl ::= '<' ! ENTITY' S EntityDeclName S EntityDef S? '>'
EntityDeclName	EntityValue ::= "" ([^%&"]   PEReference   Reference)* ""
EntityValue	EntityDeclName ::= Name
ETag	ETag ::= '<' '/' ETagName S? '>'
ETagName	ETagName ::= Name
ExternalID	ExternalID ::= 'SYSTEM' S SystemLiteral
NotationDecl	NotationDecl ::= '<' !NOTATION' S Name S (ExternalID   PublicID) S? '>'
PI	PI ::= '<' '?' PITarget (S (Char* - (Char* '?>' Char*)))? '?' '>'
PITarget	PITarget ::= Name - (('X'   'x') ('M'   'm') ('L'   'l'))
Reference	Reference ::= EntityRef   CharRef   PEReference
S Tag	S Tag ::= '<' STagName (S Attribute)* S? '>'
S TagName	S TagName ::= Name
TextDecl	TextDecl ::= '<' '?' 'xml' VersionInfo? EncodingDecl S? '?>'
XMLDecl	XMLDecl ::= '<' '?' 'xml' VersionInfo EncodingDecl? SDDDecl? S? '?>'
Methods	
token(int, String)	The interface call-back method.

---

**Fields****AttListDecl**

```
public static final int AttListDecl
AttListDecl ::= '<' ! 'ATTLIST' S Name AttDef* S? '>'
```



**AttName**

```
public static final int AttName
AttName ::= Name
```

**Attribute**

```
public static final int Attribute
Attribute ::= AttName Eq AttValue
```

**AttValue**

```
public static final int AttValue
AttValue ::= "" ([^<&" | Reference)* ""
| "" ([^<&' | Reference)* ""
```

**CDSect**

```
public static final int CDSect
CDSect ::= CDStart CData CDEnd
CDStart ::= '<' '[' [CDATA['
CData ::= (Char* - (Char* ']]>' Char*))
CDEnd ::= ']]>'
```

**CharData**

```
public static final int CharData
CharData ::= [^<&]* - ([^<&]* ']]>' [^<&]*
```

**Comment**

```
public static final int Comment
Comment ::= '<' '!' '--' ((Char - '-') | ('-' (Char - '-')))* '-->'
```

**DTDName**

```
public static final int DTDName
DTDName ::= name
```

**ElemDeclName**

```
public static final int ElemDeclName
ElemDeclName ::= name
```

**elementdecl**

```
public static final int elementdecl
elementdecl ::= '<' 'ELEMENT' S ElemDeclName S contentspec S? '>'
```

**EmptyElemTag**

```
public static final int EmptyElemTag
EmptyElemTag ::= '<' STagName (S Attribute)* S? '/' '>'
```

**EntityDecl**

```
public static final int EntityDecl
EntityDecl ::= '<' '!' ENTITY' S EntityDeclName S EntityDef S? '>'
| '<' '!' ENTITY' S '%' S EntityDeclName S PEDef S? '>'
EntityDef ::= EntityValue | (ExternalID NDataDecl?)
PEDef ::= EntityValue | ExternalID
```

**EntityDeclName**

```
public static final int EntityDeclName
EntityValue ::= "" ([^%&"] | PEReference | Reference)* ""
| "" ([^%&'] | PEReference | Reference)* ""
```

**EntityValue**

```
public static final int EntityValue
EntityDeclName ::= Name
```

**ETag**

```
public static final int ETag
ETag ::= '<' '/' ETagName S? '>'
```

**ETagName**

```
public static final int ETagName
ETagName ::= Name
```

**ExternalID**

```
public static final int ExternalID
ExternalID ::= 'SYSTEM' S SystemLiteral
```

---

| 'PUBLIC' S PubidLiteral S SystemLiteral

## NotationDecl

```
public static final int NotationDecl
NotationDecl ::= '<' '!NOTATION' S Name S (ExternalID | PublicID) S? '>'
```

## PI

```
public static final int PI
PI ::= '<' '?' PITarget (S (Char* - (Char* '?>' Char*)))? '?' '>'
```

## PITarget

```
public static final int PITarget
PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
```

## Reference

```
public static final int Reference
Reference ::= EntityRef | CharRef | PEReference
EntityRef ::= '&' Name ';'
PEReference ::= '%' Name ';'
CharRef ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
```

## STag

```
public static final int STag
STag ::= '<' STagName (S Attribute)* S? '>'
```

## STagName

```
public static final int STagName
STagName ::= Name
```

## TextDecl

```
public static final int TextDecl
TextDecl ::= '<' '?' 'xml' VersionInfo? EncodingDecl S? '?>'
```

## XMLDecl

```
public static final int XMLDecl
XMLDecl ::= '<' '?' 'xml' VersionInfo EncodingDecl? SDDecl? S? '?' '>'
```

## Methods

### **token(int, String)**

```
public void token(int token, java.lang.String value)
```

The interface call-back method. Receives an XML token and it's corresponding value

#### **Parameters**

`token` - The XML token constant as specified in the interface.

`value` - The corresponding substring from the parsed text.

# XMLTokenizer

## Syntax

```
public class XMLTokenizer extends Package oracle.xml.parser.v2 implements
oracle.xml.parser.v2.XMLConstants
```

```
java.lang.Object
|
+--org.xml.sax.HandlerBase
|
+--Package oracle.xml.parser.v2
|
+--oracle.xml.parser.v2.XMLTokenizer
```

## All Implemented Interfaces

org.xml.sax.DocumentHandler , org.xml.sax.DTDHandler , org.xml.sax.EntityResolver , org.xml.sax.ErrorHandler , oracle.xml.parser.v2.XMLConstants, XMLDocumentHandler

## Description

This class implements an eXtensible Markup Language (XML) 1.0 parser according to the World Wide Web Consortium (W3C) recommendation.

---

## Member Summary

---

### Constructors

XMLTokenizer()                      Creates a new Tokenizer object.

XMLTokenizer(XMLToken)            Creates a new Tokenizer object.

### Methods

parseDocument()                    Document ::= Prolog Element Misc\*

setErrorHandler(ErrorHandler)    Applications can use this to register a new error event handler.

setErrorStream(OutputStream)    Register a output stream for errors

setToken(int, boolean)            Applications can use this to register a new token for XML tokenizer.

setTokenHandler(XMLToken)        Applications can use this to register a new XML tokenizer event handler.

tokenize(InputSource)             Tokenizes the XML from given input source

tokenize(InputStream)             Tokenizes the XML from given input stream.

---

**Member Summary**

---

tokenize(Reader)	Tokenizes the XML from given input stream.
tokenize(String)	Tokenizes the XML from the URL indicated
tokenize(URL)	Tokenizes the XML document pointed to by the given URL and creates the corresponding XML document hierarchy.

---

---

**Inherited Member Summary**

---

Fields inherited from interface oracle.xml.parser.v2.XMLConstants

AMP, ASTERISK, cANY, cATTLIST, cCDATA, cCDATAEND, cCDATASTART, cCOMMENTEND, cCOMMENTSTART, cDECCREF, cDECLSTART, cDOCTYPE, cELEMENT, cEMPTY, cEMPTYTAGEND, cENCODING, cENDTAGSTART, cENTITIES, cENTITY, cFIXED, cHEXCREF, cID, cIDREF, cIDREFS, cIGNORE, cIMPLIED, cINCLUDE, cNDATA, cNMTOKEN, cNMTOKENS, cNOTATION, COLON, COMMA, cPIEND, cPISTART, cPUBLIC, cREQUIRED, cSTANDALONE, cSYSTEM, cVERSION, cXML, DOUBLEQUOTE, EOF, EQ, ERROR, FATAL\_ERROR, FDIGIT, FLETTER, FMISCSNAME, FSTARTNAME, FWHITESPACE, HASH, ICOUNT, ISTART, LEFTSQB, LPAREN, nameCDATA, nameCOMMENT, nameDOCUMENT, nameDOCUMENTFRAGMENT, nameENCODING, nameNameSpace, nameSpaceSeparator, nameSTANDALONE, nameTEXT, nameVERSION, nameXML, nameXMLLang, nameXMLNamespace, nameXMLNSNamespace, nameXMLSpace, nameXSLPI, NONVALIDATING, OR, PERCENT, PLUS, QMARK, QUOTE, RIGHTSQB, RPAREN, SEMICOLON, SLASH, TAGEND, TAGSTART, VALIDATING, WARNING

Methods inherited from class Package oracle.xml.parser.v2

cDATASection(char[], int, int), comment(String), endDoctype(), endElement(NSName), setDoctype(DTD), setTextDecl(String, String), setXMLDecl(String, String, String), startElement(NSName, SAXAttrList)

Methods inherited from class HandlerBase

characters(char[], int, int), endDocument(), endElement(String), error(SAXParseException), fatalError(SAXParseException), ignorableWhitespace(char[], int, int), notationDecl(String, String, String), processingInstruction(String, String), resolveEntity(String, String), setDocumentLocator(Locator), startDocument(), startElement(String, AttributeList), unparsedEntityDecl(String, String, String, String), warning(SAXParseException)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface XMLDocumentHandler

cDATASection(char[], int, int), comment(String), endDoctype(), endElement(NSName), setDoctype(DTD), setTextDecl(String, String), setXMLDecl(String, String, String), startElement(NSName, SAXAttrList)

Methods inherited from interface DocumentHandler

characters(char[], int, int), endDocument(), endElement(String), ignorableWhitespace(char[], int, int), processingInstruction(String, String), setDocumentLocator(Locator), startDocument(), startElement(String, AttributeList)

Methods inherited from interface EntityResolver

---

**Inherited Member Summary**

---

resolveEntity(String, String)

Methods inherited from interface DTDHandler

notationDecl(String, String, String), unparsedEntityDecl(String, String, String, String)

Methods inherited from interface ErrorHandler

error(SAXParseException), fatalError(SAXParseException), warning(SAXParseException)

---

**Constructors****XMLTokenizer()**

```
public XMLTokenizer()  
Creates a new Tokenizer object.
```

**XMLTokenizer(XMLToken)**

```
public XMLTokenizer(XMLToken handler)  
Creates a new Tokenizer object.
```

**Methods****parseDocument()**

```
public void parseDocument()  
Document ::= Prolog Element Misc*
```

**setErrorHandler(ErrorHandler)**

```
public void setErrorHandler(org.xml.sax.ErrorHandler handler)  
Applications can use this to register a new error event handler. This replaces any  
previous setting for error handling.
```

**Parameters**

handler - ErrorHandler being registered

**setErrorStream(OutputStream)**

```
public void setErrorStream(java.io.OutputStream out)  
Register a output stream for errors
```

### **setToken(int, boolean)**

```
public void setToken(int token, boolean val)
```

Applications can use this to register a new token for XML tokenizer.

#### **Parameters**

token - XMLToken being set

### **setTokenHandler(XMLToken)**

```
public void setTokenHandler(XMLToken handler)
```

Applications can use this to register a new XML tokenizer event handler.

#### **Parameters**

handler - XMLToken being registered

### **tokenize(InputSource)**

```
public final void tokenize(org.xml.sax.InputSource in)
```

Tokenizes the XML from given input source

#### **Parameters**

in - the org.xml.sax.InputSource to parse

#### **Throws**

XMLParseException - if syntax or other error encountered.

org.xml.sax.SAXException - Any SAX exception, possibly wrapping another exception.

IOException - IO Error.

### **tokenize(InputStream)**

```
public final void tokenize(java.io.InputStream in)
```

Tokenizes the XML from given input stream.

#### **Parameters**

in - the InputStream containing XML data to parse.

#### **Throws**

XMLParseException - if syntax or other error encountered.



`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

### See Also

`setBaseURL(URL)`

## `tokenize(Reader)`

```
public final void tokenize(java.io.Reader r)
```

Tokenizes the XML from given input stream.

### Parameters

`r` - the `Reader` containing XML data to parse.

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

### See Also

`setBaseURL(URL)`

## `tokenize(String)`

```
public final void tokenize(java.lang.String in)
```

Tokenizes the XML from the URL indicated

### Parameters

`in` - the `String` containing the URL to parse from

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

## tokenize(URL)

```
public final void tokenize(java.net.URL url)
```

Tokenizes the XML document pointed to by the given URL and creates the corresponding XML document hierarchy.

### Parameters

`url` - the url points to the XML document to parse.

### Throws

`XMLParseException` - if syntax or other error encountered.

`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

`IOException` - IO Error.

# XSLException

## Syntax

```
public class XSLException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--oracle.xml.parser.v2.XSLException
```

## All Implemented Interfaces

```
java.io.Serializable
```

## Description

Indicates that an exception occurred during XSL transformation

---

## Inherited Member Summary

---

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

## XSLProcessor

### Syntax

```
public class XSLProcessor extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.parser.v2.XSLProcessor
```

### Description

This class provides methods to transform an input XML document using a previously constructed `XSLStyleSheet`. The transformation effected is as specified by the XSLT 1.0 specification.

---

### Member Summary

---

#### Constructors

`XSLProcessor()`

#### Methods

<code>processXSL(XSLStyleSheet, InputStream, URL)</code>	Transform input XML document using given <code>InputStream</code> and stylesheet.
<code>processXSL(XSLStyleSheet, Reader, URL)</code>	Transform input XML document using given <code>Reader</code> and stylesheet.
<code>processXSL(XSLStyleSheet, URL, URL)</code>	Transform input XML document using given <code>URL</code> and stylesheet.
<code>processXSL(XSLStyleSheet, XMLDocument)</code>	Transform input XML document using given <code>XMLDocument</code> and stylesheet.
<code>processXSL(XSLStyleSheet, XMLDocumentFragment)</code>	Transform input XML document using given <code>XMLDocument</code> and stylesheet.
<code>processXSL(XSLStyleSheet, XMLDocumentFragment, OutputStream)</code>	Transform input XML using given <code>XMLDocumentFragment</code> and stylesheet.
<code>processXSL(XSLStyleSheet, XMLDocumentFragment, PrintWriter)</code>	Transform input XML using given <code>XMLDocumentFragment</code> and stylesheet.
<code>processXSL(XSLStyleSheet, XMLDocument, OutputStream)</code>	Transform input XML document using given <code>XMLDocument</code> and stylesheet.

---

**Member Summary**

---

<code>processXSL(XSLStylesheet, XMLDocument, PrintWriter)</code>	Transform input XML document using given XMLDocument and stylesheet.
<code>setErrorStream(OutputStream)</code>	Creates an output stream for the output of warnings.
<code>setLocale(Locale)</code>	Applications can use this to set the locale for error reporting.
<code>showWarnings(boolean)</code>	Switch to determine whether to output warnings.

---

---

**Inherited Member Summary**

---

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

**Constructor****XSLProcessor()**

```
public XSLProcessor()
```

**Methods****processXSL(XSLStylesheet, InputStream, URL)**

```
public XMLDocumentFragment processXSL(XSLStylesheet xsl, java.io.InputStream xml, java.net.URL ref)
```

Transforms input XML document using given InputStream and stylesheet.

**Parameters**

`xsl` - XSLStylesheet to be used for transformation

`xml` - XML input to be transformed (as a java.io.Inputstream)

`ref` - Reference URL to resolve external entities in input xml file

**Returns**

XMLDocumentFragment

**Throws**

XSLException - on error.

### **processXSL(XSLStylesheet, Reader, URL)**

```
public XMLDocumentFragment processXSL(XSLStylesheet xsl, java.io.Reader xml,  
java.net.URL ref)
```

Transform input XML document using given Reader and stylesheet.

#### **Parameters**

xsl - XSLStylesheet to be used for transformation

xml - XML input to be transformed (as a java.io.Reader)

ref - Reference URL to resolve external entities in input xml file

#### **Returns**

XMLDocumentFragment

#### **Throws**

XSLException - on error.

### **processXSL(XSLStylesheet, URL, URL)**

```
public XMLDocumentFragment processXSL(XSLStylesheet xsl, java.net.URL xml,  
java.net.URL ref)
```

Transform input XML document using given URL and stylesheet.

#### **Parameters**

xsl - XSLStylesheet to be used for transformation

xml - XML input to be transformed (as a java.net.URL)

ref - Reference URL to resolve external entities in input xml file

#### **Returns**

XMLDocumentFragment

#### **Throws**

XSLException - on error.

### **processXSL(XSLStylesheet, XMLDocument)**

```
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocument xml)
```

Transform input XML document using given XMLDocument and stylesheet.

**Parameters**

xsl - XSLStylesheet to be used for transformation

xml - XML input to be transformed (as a DOM Tree)

**Returns**

XMLDocumentFragment

**Throws**

XSLException - on error.

**processXSL(XSLStylesheet, XMLDocumentFragment)**

```
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocumentFragment  
inp)
```

Transform input XML document using given XMLDocument and stylesheet.

**Parameters**

xsl - XSLStylesheet to be used for transformation

xml - XML input to be transformed (as a DOM Tree)

**Returns**

XMLDocumentFragment

**Throws**

XSLException - on error.

**processXSL(XSLStylesheet, XMLDocumentFragment, OutputStream)**

```
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml,  
java.io.OutputStream out)
```

Transform input XML using given XMLDocumentFragment and stylesheet.

**Parameters**

xsl - XSLStylesheet to be used for transformation

xml - XML input to be transformed (as a DOM Tree)

out - Outputstream to which the result is printed

### Throws

`XSLException`, - `IOException` on error.

### `processXSL(XSLStylesheet, XMLDocumentFragment, PrintWriter)`

```
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml,  
    java.io.PrintWriter pw)
```

Transform input XML using given `XMLDocumentFragment` and stylesheet.

### Parameters

`xsl` - `XSLStylesheet` to be used for transformation

`xml` - XML input to be transformed (as a DOM Tree)

`pw` - `PrintWriter` to which the result is printed

### Throws

`XSLException`, - `IOException` on error.

### `processXSL(XSLStylesheet, XMLDocument, OutputStream)`

```
public void processXSL(XSLStylesheet xsl, XMLDocument xml, java.io.OutputStream  
    out)
```

Transform input XML document using given `XMLDocument` and stylesheet.

### Parameters

`xsl` - `XSLStylesheet` to be used for transformation

`xml` - XML input to be transformed (as a DOM Tree)

`out` - `OutputStream` to which the result is printed

### Throws

`XSLException`, - `IOException` on error.

### `processXSL(XSLStylesheet, XMLDocument, PrintWriter)`

```
public void processXSL(XSLStylesheet xsl, XMLDocument xml, java.io.PrintWriter  
    pw)
```

Transform input XML document using given `XMLDocument` and stylesheet.

### Parameters

`xsl` - `XSLStylesheet` to be used for transformation



`xml` - XML input to be transformed (as a DOM Tree)

`pw` - `PrintWriter` to which the result is printed

### Throws

`XSLException`, - `IOException` on error.

### **setErrorStream(OutputStream)**

```
public final void setErrorStream(java.io.OutputStream out)
```

Creates an output stream for the output of warnings. If an output stream for warnings is not specified, the processor will not output any warnings

### Parameters

`out` - The output stream to use for errors and warnings

### **setLocale(Locale)**

```
public void setLocale(java.util.Locale locale)
```

Applications can use this to set the locale for error reporting.

### Parameters

`locale` - `Locale` to set

### **showWarnings(boolean)**

```
public final void showWarnings(boolean yes)
```

Switch to determine whether to output warnings.

### Parameters

`yes` - determines whether warnings should be shown By default, warnings are not output

## XSLStylesheet

### Syntax

```
public class XSLStylesheet extends java.lang.Object implements
oracle.xml.parser.v2.XSLConstants
```

```
java.lang.Object
|
+--oracle.xml.parser.v2.XSLStylesheet
```

### All Implemented Interfaces

```
oracle.xml.parser.v2.XSLConstants
```

### Description

The class holds XSL stylesheet information such as templates, keys, variables, and attribute sets. The same stylesheet, once constructed, can be used to transform multiple XML documents.

---

### Member Summary

---

#### Constructors

XSLStylesheet(InputStream, URL)	Constructs an XSLStylesheet using the given Inputstream
XSLStylesheet(Reader, URL)	Constructs an XSLStylesheet using the given Reader
XSLStylesheet(URL, URL)	Constructs an XSLStylesheet using the given URL
XSLStylesheet(XMLDocument, URL)	Constructs an XSLStylesheet using the given XMLDocument

#### Methods

setParam(String, String)	Sets the value of a top-level stylesheet parameter.
--------------------------	---

---

---

### Inherited Member Summary

---

Fields inherited from interface oracle.xml.parser.v2.XSLConstants

APPLY\_IMPORTS, APPLY\_TEMPLATES, ATTRIBUTE, ATTRIBUTE\_SET, CALL\_TEMPLATE, CHOOSE, COMMENT, COPY, COPY\_OF, DISABLEOUTESC, ELEMENT, FOR\_EACH, HREF, IF, IMPORT, INCLUDE, KEY, LOCALE, MATCH, MESSAGE, NAME, NEGINFPRIORITY, NUMBER, ORACLE\_NAME, ORACLE\_URL, OTHERWISE, OUTPUT, PARAM, PARAM\_VARIABLE, PI, PRESERVE\_SPACE, RESULT\_ROOT, SORT, STRIP\_SPACE, TEMPLATE, TEXT, USE, USE\_ATTRIBUTE\_SETS, VALUE\_OF, VARIABLE, WHEN, XSL\_ROOT, XSLEXTFUNCNS, XSLNAMESPACE, XSLT\_SPEC\_VERSION

---

**Inherited Member Summary**

---

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

**Constructors****XSLStylesheet(InputStream, URL)**

```
public XSLStylesheet(java.io.InputStream xsl, java.net.URL ref)
Constructs an XSLStylesheet using the given Inputstream
```

**Parameters**

`xsl` - XSL input as an Inputstream

`ref` - Reference URL for include, import and external entities

**Throws**

XSLException - on error.

**XSLStylesheet(Reader, URL)**

```
public XSLStylesheet(java.io.Reader xsl, java.net.URL ref)
Constructs an XSLStylesheet using the given Reader
```

**Parameters**

`xsl` - XSL input as a Reader

`ref` - Reference URL for include, import and external entities

**Throws**

XSLException - on error.

**XSLStylesheet(URL, URL)**

```
public XSLStylesheet(java.net.URL xsl, java.net.URL ref)
Constructs an XSLStylesheet using the given URL
```

**Parameters**

`xsl` - XSL input as a URL

`ref` - Reference URL for include, import and external entities

### Throws

XSLException - on error.

### XSLStylesheet(XMLDocument, URL)

```
public XSLStylesheet(XMLDocument xsl, java.net.URL ref)
```

Constructs an XSLStylesheet using the given XMLDocument

### Parameters

`xsl` - XSL input as a DOM Tree

`ref` - Reference URL for include, import

### Throws

XSLException - on error.

## Methods

### setParam(String, String)

```
public void setParam(java.lang.String name, java.lang.String value)
```

Sets the value of a top-level stylesheet parameter. The parameter value is expected to be a valid XPath expression (note that string literal values would therefore have to be explicitly quoted).

### Parameters

`name` - parameter name

`value` - parameter value as an XPath expression

### Throws

XSLException - on error

---

## Package oracle.xml.classgen

## CGDocument

### Syntax

```
public abstract class CGDocument extends oracle.xml.classgen.CGNode
```

```
java.lang.Object  
|  
+--oracle.xml.classgen.CGNode  
|  
+--oracle.xml.classgen.CGDocument
```

### Description

Serves as the base document class for the DTD compiler generated classes

## Constructors

### CGDocument()

```
protected CGDocument()
```

### CGDocument(String, DTD)

```
protected CGDocument(java.lang.String doctype, oracle.xml.parser.v2.DTD dtd)
```

Constructor for the Root element of the DTD

### Parameters:

`doctype` - Name of the root Element of the DTD

`dtd` - The DTD used to generate the classes

## Methods

### print(OutputStream)

```
protected void print(java.io.OutputStream out)
```

Prints the constructed XML Document

**Parameters:**

out - Output stream to which the document will be printed

**Throws:**

`InvalidContentException` - Throw exception if the document's content do not match the grammer specified by DTD (The validation mode should be set to TRUE)

**See Also:**

`oracle.xml.classgen.ClassGenerator`

**print(OutputStream, String)**

```
protected void print(java.io.OutputStream out, java.lang.String enc)
```

Prints the constructed XML Document

**Parameters:**

out - Output stream to which the document will be printed

enc - Encoding of the output stream

**Throws:**

`InvalidContentException` - Throw exception if the document's content do not match the grammer specified by DTD (The validation mode should be set to TRUE)

**See Also:**

`oracle.xml.classgen.ClassGenerator`

## CGNode

### Syntax

```
public abstract class CGNode extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.classgen.CGNode
```

### Direct Known Subclasses:

CGDocument

### Description

Serves as the base class for nodes generated by the DTD compiler

### Fields

#### isValidating

```
protected boolean isValidating  
Boolean to indicate the validating mode
```

### Constructors

#### CGNode()

#### CGNode(String)

```
protected CGNode(java.lang.String elementName)  
Constructor for the Elements of the DOM Tree
```

#### Parameters:

elementName - Name of the element



## Methods

### addCDATASection(String)

protected void addCDATASection(java.lang.String theData)

Adds CDATA Section to the Element

#### Parameters:

theData - Text to be added as CDATA Section to the element

#### Throws:

InvalidContentException - Thrown if theData has illegal characters (validation must be set to TRUE)

#### See Also:

oracle.xml.classgen.ClassGenerator

### addData(String)

protected void addData(java.lang.String theData)

Adds PCDATA to the Element

#### Parameters:

theData - Text to be added to the element

#### Throws:

InvalidContentException - Thrown if theData has illegal characters (validation must be set to TRUE)

#### See Also:

oracle.xml.classgen.ClassGenerator

### addNode(CGNode)

protected void addNode(CGNode theNode)

Adds a node as a child to the element

#### Parameters:

theNode - The node to be added as child

**Throws:**

`InvalidContentException` - Thrown if the Node cannot be added as child as per Content Model of the element (validation must be set to TRUE)

**See Also:**

`oracle.xml.classgen.ClassGenerator`

**getCGDocument()**

`protected CGDocument getCGDocument()`  
Gets the base document (root Element)

**Returns:**

The base CGDocument

**getDTDNode()**

`protected abstract oracle.xml.parser.v2.DTD getDTDNode()`  
Gets the static DTD from the base document

**Returns:**

DTD stored in base CGDocument

**setAttribute(String, String)**

`protected void setAttribute(java.lang.String attName, java.lang.String value)`  
Sets the value of the Attribute

**Parameters:**

`attName` - Name of the attribute

`value` - Value of the attribute

**setDocument(CGDocument)**

`public void setDocument(CGDocument d)`  
Sets the base document (root Element)

**Parameters:**

`d` - Base CGDocument

**storeID(String, String)**

protected void storeID(java.lang.String attName, java.lang.String id)  
Store this value for an ID identifier, so that we can later verify IDREF values

**Parameters:**

attName - Name of the ID Attribute

id - Value of the ID

**storeIDREF(String, String)**

protected void storeIDREF(java.lang.String attName, java.lang.String idref)  
Store this value for an IDREF identifier, so that we can later verify, if an corresponding ID was defined.

**Parameters:**

attName - Name of the IDREF Attribute

idref - Value of the IDREF

**validateContent()**

protected void validateContent()  
Checks if the content of the element is valid as per the Content Model specified in DTD

**Returns:**

True if content is valid, else false

**validEntity(String)**

protected boolean validEntity(java.lang.String entity)  
Checks if the ENTITY identifier is valid

**Parameters:**

name - value of the Entity Attribute

**Returns:**

True if Entity is valid, else false

### **validID(String)**

protected boolean validID(java.lang.String name)  
Checks if the ID identifier is valid

**Parameters:**

name - value of the ID Attribute

**Returns:**

True if ID is valid, else false

### **validNMTOKEN(String)**

protected boolean validNMTOKEN(java.lang.String name)  
Checks if the NMTOKEN identifier is valid

**Parameters:**

name - value of the Nmtoken Attribute

**Returns:**

True if Nmtoken is valid, else false

## CGXSDElement

### Syntax

```
public abstract class CGXSDElement
```

```
oracle.xml.classgen.CGXSDElement
```

### Description

This class serves as the base class for the all the generated classes corresponding to the XML Schema generated by Schema Class Generator

### Fields

#### type

```
protected java.lang.Object type
```

### Constructors

#### CGXSDElement()

```
public CGXSDElement()
```

### Methods

#### addAttribute(String, String)

```
protected void addAttribute(java.lang.String attName, java.lang.String attValue)
```

Add the attribute of a given node to the hashtable.

#### Parameters:

attName - the attribute name

attValue - the attribute value

#### addElement(Object)

```
protected void addElement(java.lang.Object elem)
```

Add the elements of a given element node to the vector correspondig to the elements.

**Parameters:**

elem - the object which needs to be added

**getAttributes()**

```
public java.util.Hashtable getAttributes()  
Return the attributes
```

**Returns:**

attributes the hashtable containing attribute name and value

**getChildElements()**

```
public java.util.Vector getChildElements()  
Get the vector having all the local elements
```

**Returns:**

elemChild vector

**getNodeValue()**

```
public java.lang.String getNodeValue()  
Return the node value
```

**getType()**

```
public java.lang.Object getType()  
Return the type
```

**print(XMLOutputStream)**

```
public void print(oracle.xml.parser.v2.XMLOutputStream out)  
Print an element node
```

**Parameters:**

out - the stream where the output is printed

**printAttributes(XMLOutputStream, String)**

```
public void printAttributes(oracle.xml.parser.v2.XMLOutputStream out,
```

java.lang.String name)

Print an attribute node

**Parameters:**

out - the stream where the output is printed

name - the attribute name

**setNodeValue(String)**

protected void setNodeValue(java.lang.String value)

Set the node value of an element

**Parameters:**

value - the node value

## DTDClassGenerator

### Syntax

```
public class DTDClassGenerator extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.classgen.DTDClassGenerator
```

### Description

This class is used by the DTD compiler to generate classes

### Constructors

#### DTDClassGenerator()

```
public DTDClassGenerator()  
Default constructor for DTDClassGenerator.
```

### Methods

#### generate(DTD, String)

```
public void generate(oracle.xml.parser.v2.DTD dtd, java.lang.String doctype)  
Traverses the DTD with element doctype as root and generates Java classes
```

#### Parameters:

DTD - The DTD used to generate the classes

doctype - Name of the root Element

#### printDocumentMethods()

```
protected void printDocumentMethods()
```

#### setGenerateComments(boolean)

```
public void setGenerateComments(boolean comments)
```



Switch to determine whether to generate java doc comments Default - TRUE

**Parameters:**

`comments` - boolean flag

**setJavaPackage(Vector)**

`public void setJavaPackage(java.util.Vector packageName)`  
Sets the package for the classes generated Default - No package

**Parameters:**

`packageName` - Name of the package

**setOutputDirectory(String)**

`public void setOutputDirectory(java.lang.String dir)`  
Sets the output directory where the java source code for the DTD are generated.  
Default - current directory

**Parameters:**

`dir` - Output directory

**setSerializationMode(boolean)**

`public void setSerializationMode(boolean yes)`  
Switch to determine if the DTD should be saved as a serialized object or as text file.  
Serializing the DTD improves the performance when the generated classes are used  
to author XML files. Default - FALSE (DTD is saved a text file)

**Parameters:**

`yes` - boolean flag

**setValidationMode(boolean)**

`public void setValidationMode(boolean yes)`  
Switch to determine whether the classes generated should validate the XML  
Document being constructed Default - TRUE

**Parameters:**

`yes` - boolean flag

## InvalidContentException

### Syntax

```
public class InvalidContentException extends java.lang.Exception
```

```
java.lang.Object  
|  
+--java.lang.Throwable  
|  
+--java.lang.Exception  
|  
+--oracle.xml.classgen.InvalidContentException
```

### All Implemented Interfaces:

```
java.io.Serializable
```

### Description

Definition of InvalidContentException thrown by dtdcompiler classes

### Constructors

#### InvalidContentException()

```
public InvalidContentException()
```

#### InvalidContentException(String)

```
public InvalidContentException(java.lang.String s)
```

## oracg

### Syntax

```
public class oracg extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.classgen.oracg
```

### Description

The oracg class provides a command-line interface to generate java classes corresponding to the DTD or XML Schema java oracle.xml.classgen.oracg options are: -h Prints the help message text -d The input file is a DTD file or DTD based XML file -s The input file is a Schema file or Schema based XML file -o The directory name where java source is generated -p The package name(s) of the generated java classes. -c Generate comments for the generated java source code.

## Constructors

### oracg()

```
public oracg()  
Default constructor for oracg
```

## Methods

### main(String[])

```
public static void main(java.lang.String[] args)  
The main method
```

## SchemaClassGenerator

### Syntax

```
public class SchemaClassGenerator extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+-oracle.xml.classgen.SchemaClassGenerator
```

### Description

This class generates the classes corresponding to an XML Schema.

## Constructors

### SchemaClassGenerator()

```
public SchemaClassGenerator()
```

Default empty constructor for Schema Class Generator

### SchemaClassGenerator(String)

```
public SchemaClassGenerator(java.lang.String fileName)
```

The constructor for Schema Class Generator

### Parameters:

fileName - the input XML Schema

## Methods

### generate(XMLSchema)

```
public void generate(oracle.xml.parser.schema.XMLSchema schema)
```

Generate the Schema classes corresponding to top level elements, simpleType elements and complexType elements by calling createSchemaClass on each of these nodes.

**Parameters:**

XML - Schema object

**Throws:****setGenerateComments(boolean)**

```
public void setGenerateComments(boolean comments)
```

Switch to determine whether to generate java doc comments The default setting is true

**Parameters:**

comments - boolean flag

**setJavaPackage(XMLSchema, Vector)**

```
public void setJavaPackage(oracle.xml.parser.schema.XMLSchema schema,  
java.util.Vector pkgName)
```

Sets the Java package names corresponding to Namespaces. The Namespaces defined in the schema are queried. The Number of namespaces defined in the schema should match the number of package names given by the user through command line else an error is thrown. For each namespace, a user-defined package name is assigned.

**Parameters:**

schema - XMLSchema

pkgName - A vector containing user defined package names given through command line.

**setOutputDirectory(String)**

```
public void setOutputDirectory(java.lang.String dir)
```

Sets the output directory where the java source code for the Schema class are generated. Default - current directory

**Parameters:**

dir - Output directory



---

## Package oracle.xml.xsql

This package is also known as the XSQL Servlet.

## Description

---

### Class Summary

---

#### Interfaces

[XSQActionHandler](#) Interface that must be implemented by all XSQL Action Element Handlers

[XSQLPageRequest](#) Interface representing a request for an XSQL Page

#### Classes

#### Res

[XSQLActionHandlerImpl](#) Base Implementation of XSQLActionHandler that can be extended to create your own custom handlers.

[XSQLCommandLine](#) Command-line Utility to process XSQL Pages.

[XSQLDiagnostic](#)

[XSQLHttpUtil](#)

[XSQLPageRequestImpl](#) Base implementation of the XSQLPageRequest interface that can be used to derive new kinds of page request implementations.

[XSQLParserHelper](#) Common XML Parsing Routines

[XSQLRequest](#) Programmatically process a request for an XSQL Page.

[XSQLServlet](#) Servlet to enable HTTP GET-ing of and POST-ing to XSQL Pages

[XSQLServletPageRequest](#) Implementation of XSQLPageRequest for Servlet-based XSQL Page requests.

[XSQLStylesheetProcessor](#) XSLT Stylesheet Processing Engine

[XSQLUtil](#)

---



# Res

## Syntax

```
public class Res extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.Res
```

## Description

---

### Member Summary

---

#### Fields

[CANTREAD\\_XSQL](#)  
[CANTREAD\\_XSQL\\_MSG](#)  
[CLASSNOTFOUND](#)  
[CLASSNOTFOUND\\_MSG](#)  
[CONN\\_FILE](#)  
[CONN\\_FILE\\_MSG](#)  
[ERR\\_OUTPUT](#)  
[ERR\\_OUTPUT\\_MSG](#)  
[ERRORINCLUDING](#)  
[ERRORINCLUDING\\_MSG](#)  
[ERRORLOADINGURL](#)  
[ERRORLOADINGURL\\_MSG](#)  
[ERRORREADINGPARAM](#)  
[ERRORREADINGPARAM\\_MSG](#)  
[FATAL\\_SHEETPOOL](#)  
[FATAL\\_SHEETPOOL\\_MSG](#)  
[ILLFORMEDXMLPARAMVAL](#)  
[ILLFORMEDXMLPARAMVAL\\_MSG](#)

---

### **Member Summary**

---

ILLFORMEDXMLRESOURCE  
ILLFORMEDXMLRESOURCE\_MSG  
INSTANTIATIONERR  
INSTANTIATIONERR\_MSG  
INVALID\_URI  
INVALID\_URI\_MSG  
INVALIDURL  
INVALIDURL\_MSG  
MISSING\_ARGS  
MISSING\_ARGS\_MSG  
MISSING\_ATTR  
MISSING\_ATTR\_MSG  
NAMED\_CONN  
NAMED\_CONN\_MSG  
NO\_CONN  
NO\_CONN\_DEF  
NO\_CONN\_DEF\_MSG  
NO\_CONN\_MSG  
NO\_XSQL\_FILE  
NO\_XSQL\_FILE\_MSG  
NOFUNCTIONNAME  
NOFUNCTIONNAME\_MSG  
NOPOSTEDXML  
NOPOSTEDXML\_MSG  
NOQUERYSUPPLIED  
NOQUERYSUPPLIED\_MSG  
NOTANACTIONHANDLER  
NOTANACTIONHANDLER\_MSG

---

**Member Summary**

---

NULLPARAM

NULLPARAM\_MSG

OWAXMLMALFORMED

OWAXMLMALFORMED\_MSG

POSTEDXML\_ERR

POSTEDXML\_ERR\_MSG

UNHANDLED\_ERR

UNHANDLED\_ERR\_MSG

UNHANDLED\_ERR\_XSL\_PR

UNHANDLED\_ERR\_XSL\_PR\_MSG

UNHANDLED\_ERR\_XSL\_RD

UNHANDLED\_ERR\_XSL\_RD\_MSG

XML\_INS\_ERR

XML\_INS\_ERR\_MSG

XML\_PARSE

XML\_PARSE\_MSG

XML\_SQL\_ERR

XML\_SQL\_ERR\_MSG

XSL\_ERRORS

XSL\_ERRORS\_MSG

XSL\_NOFILE

XSL\_NOFILE\_MSG

XSL\_PARSE

XSL\_PARSE\_MSG

XSLNOTFOUND

XSLNOTFOUND\_MSG

**Constructors**

Res()

---

### Member Summary

---

#### Methods

[format\(int, String\)](#)

[getString\(int\)](#)

---

---

### Inherited Member Summary

---

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

---

### Fields

#### CANTREAD\_XSQL

```
public static final int CANTREAD_XSQL
```

#### CANTREAD\_XSQL\_MSG

```
public static final java.lang.String CANTREAD_XSQL_MSG
```

#### CLASSNOTFOUND

```
public static final int CLASSNOTFOUND
```

#### CLASSNOTFOUND\_MSG

```
public static final java.lang.String CLASSNOTFOUND_MSG
```

#### CONN\_FILE

```
public static final int CONN_FILE
```

#### CONN\_FILE\_MSG

```
public static final java.lang.String CONN_FILE_MSG
```

#### ERR\_OUTPUT

```
public static final int ERR_OUTPUT
```

**ERR\_OUTPUT\_MSG**

```
public static final java.lang.String ERR_OUTPUT_MSG
```

**ERRORINCLUDING**

```
public static final int ERRORINCLUDING
```

**ERRORINCLUDING\_MSG**

```
public static final java.lang.String ERRORINCLUDING_MSG
```

**ERRORLOADINGURL**

```
public static final int ERRORLOADINGURL
```

**ERRORLOADINGURL\_MSG**

```
public static final java.lang.String ERRORLOADINGURL_MSG
```

**ERRORREADINGPARAM**

```
public static final int ERRORREADINGPARAM
```

**ERRORREADINGPARAM\_MSG**

```
public static final java.lang.String ERRORREADINGPARAM_MSG
```

**FATAL\_SHEETPOOL**

```
public static final int FATAL_SHEETPOOL
```

**FATAL\_SHEETPOOL\_MSG**

```
public static final java.lang.String FATAL_SHEETPOOL_MSG
```

**ILLFORMEDXMLPARAMVAL**

```
public static final int ILLFORMEDXMLPARAMVAL
```

**ILLFORMEDXMLPARAMVAL\_MSG**

```
public static final java.lang.String ILLFORMEDXMLPARAMVAL_MSG
```

**ILLFORMEDXMLRESOURCE**

```
public static final int ILLFORMEDXMLRESOURCE
```

## **ILLFORMEDXMLRESOURCE\_MSG**

```
public static final java.lang.String ILLFORMEDXMLRESOURCE_MSG
```

## **INSTANTIATIONERR**

```
public static final int INSTANTIATIONERR
```

## **INSTANTIATIONERR\_MSG**

```
public static final java.lang.String INSTANTIATIONERR_MSG
```

## **INVALID\_URI**

```
public static final int INVALID_URI
```

## **INVALID\_URI\_MSG**

```
public static final java.lang.String INVALID_URI_MSG
```

## **INVALIDURL**

```
public static final int INVALIDURL
```

## **INVALIDURL\_MSG**

```
public static final java.lang.String INVALIDURL_MSG
```

## **MISSING\_ARGS**

```
public static final int MISSING_ARGS
```

## **MISSING\_ARGS\_MSG**

```
public static final java.lang.String MISSING_ARGS_MSG
```

## **MISSING\_ATTR**

```
public static final int MISSING_ATTR
```

## **MISSING\_ATTR\_MSG**

```
public static final java.lang.String MISSING_ATTR_MSG
```

## **NAMED\_CONN**

```
public static final int NAMED_CONN
```

**NAMED\_CONN\_MSG**

```
public static final java.lang.String NAMED_CONN_MSG
```

**NO\_CONN**

```
public static final int NO_CONN
```

**NO\_CONN\_DEF**

```
public static final int NO_CONN_DEF
```

**NO\_CONN\_DEF\_MSG**

```
public static final java.lang.String NO_CONN_DEF_MSG
```

**NO\_CONN\_MSG**

```
public static final java.lang.String NO_CONN_MSG
```

**NO\_XSQL\_FILE**

```
public static final int NO_XSQL_FILE
```

**NO\_XSQL\_FILE\_MSG**

```
public static final java.lang.String NO_XSQL_FILE_MSG
```

**NOFUNCTIONNAME**

```
public static final int NOFUNCTIONNAME
```

**NOFUNCTIONNAME\_MSG**

```
public static final java.lang.String NOFUNCTIONNAME_MSG
```

**NOPOSTEDXML**

```
public static final int NOPOSTEDXML
```

**NOPOSTEDXML\_MSG**

```
public static final java.lang.String NOPOSTEDXML_MSG
```

**NOQUERYSUPPLIED**

```
public static final int NOQUERYSUPPLIED
```

## **NOQUERYSUPPLIED\_MSG**

```
public static final java.lang.String NOQUERYSUPPLIED_MSG
```

## **NOTANACTIONHANDLER**

```
public static final int NOTANACTIONHANDLER
```

## **NOTANACTIONHANDLER\_MSG**

```
public static final java.lang.String NOTANACTIONHANDLER_MSG
```

## **NULLPARAM**

```
public static final int NULLPARAM
```

## **NULLPARAM\_MSG**

```
public static final java.lang.String NULLPARAM_MSG
```

## **OWAXMLMALFORMED**

```
public static final int OWAXMLMALFORMED
```

## **OWAXMLMALFORMED\_MSG**

```
public static final java.lang.String OWAXMLMALFORMED_MSG
```

## **POSTEDXML\_ERR**

```
public static final int POSTEDXML_ERR
```

## **POSTEDXML\_ERR\_MSG**

```
public static final java.lang.String POSTEDXML_ERR_MSG
```

## **UNHANDLED\_ERR**

```
public static final int UNHANDLED_ERR
```

## **UNHANDLED\_ERR\_MSG**

```
public static final java.lang.String UNHANDLED_ERR_MSG
```

## **UNHANDLED\_ERR\_XSL\_PR**

```
public static final int UNHANDLED_ERR_XSL_PR
```



**UNHANDLED\_ERR\_XSL\_PR\_MSG**

```
public static final java.lang.String UNHANDLED_ERR_XSL_PR_MSG
```

**UNHANDLED\_ERR\_XSL\_RD**

```
public static final int UNHANDLED_ERR_XSL_RD
```

**UNHANDLED\_ERR\_XSL\_RD\_MSG**

```
public static final java.lang.String UNHANDLED_ERR_XSL_RD_MSG
```

**XML\_INS\_ERR**

```
public static final int XML_INS_ERR
```

**XML\_INS\_ERR\_MSG**

```
public static final java.lang.String XML_INS_ERR_MSG
```

**XML\_PARSE**

```
public static final int XML_PARSE
```

**XML\_PARSE\_MSG**

```
public static final java.lang.String XML_PARSE_MSG
```

**XML\_SQL\_ERR**

```
public static final int XML_SQL_ERR
```

**XML\_SQL\_ERR\_MSG**

```
public static final java.lang.String XML_SQL_ERR_MSG
```

**XSL\_ERRORS**

```
public static final int XSL_ERRORS
```

**XSL\_ERRORS\_MSG**

```
public static final java.lang.String XSL_ERRORS_MSG
```

**XSL\_NOFILE**

```
public static final int XSL_NOFILE
```

## **XSL\_NOFILE\_MSG**

```
public static final java.lang.String XSL_NOFILE_MSG
```

## **XSL\_PARSE**

```
public static final int XSL_PARSE
```

## **XSL\_PARSE\_MSG**

```
public static final java.lang.String XSL_PARSE_MSG
```

## **XSLNOTFOUND**

```
public static final int XSLNOTFOUND
```

## **XSLNOTFOUND\_MSG**

```
public static final java.lang.String XSLNOTFOUND_MSG
```

## **Constructors**

### **Res()**

```
public Res()
```

## **Methods**

### **format(int, String)**

```
public static java.lang.String format(int id, java.lang.String s)
```

### **getString(int)**

```
public static java.lang.String getString(int id)
```

# XSQLActionHandler

## Syntax

```
public interface XSQLActionHandler
```

## All Known Implementing Classes:

[XSQLActionHandlerImpl](#)

## Description

Interface that must be implemented by all XSQL Action Element Handlers

Upon encountering an XSQL Action Element of the form <xsql:xxxx> in an XSQL page, the XSQL Page Processor invokes the associated XSQL Action Handler by:

1. Constructing an instance of the handler using the no-args constructor

## Invoking the XSQL Action Handler's `handleAction()` method

**NOTE:** conn parameter can be null if no connection specified for the XSQL page being processed.

---

### Member Summary

---

#### Methods

<a href="#">handleAction(Node)</a>	Handle the action, typically by executing some code and appending new child DOM nodes to the rootNode.
<a href="#">init(XSQLPageRequest, Element)</a>	Initialize the Action Handler

---

## Methods

### handleAction(Node)

```
public void handleAction(oracle.xml.xsql.Node rootNode)
```

Handle the action, typically by executing some code and appending new child DOM nodes to the rootNode.

The XSQL Page Processor replaces the action element in the XSQL Page being processed with the document fragment of nodes that your `handleAction` method appends to the `rootNode`.

**Parameters:**

`rootNode` - Root node of generated document fragment

**init(XSQLPageRequest, Element)**

```
public void init(XSQLPageRequest env, oracle.xml.xsql.Element e)  
Initialize the Action Handler
```

**Parameters:**

`env` - XSQLPageRequest object

`e` - DOM element representing the Action Element being handled

---

# XSQLActionHandlerImpl

## Syntax

public abstract class XSQLActionHandlerImpl extends java.lang.Object implements [XSQLActionHandler](#)

```
java.lang.Object
|
+--oracle.xml.xsql.XSQLActionHandlerImpl
```

## All Implemented Interfaces:

[XSQLActionHandler](#)

## Description

Base Implementation of XSQLActionHandler that can be extended to create your own custom handlers.

Includes a set of useful helper methods.

**NOTE:** If you extend this class and override the `init()` method, make sure to call:

```
super.init(env,e);
```

---

## Member Summary

---

Constructors

[XSQLActionHandlerImpl\(\)](#)

Methods

[init\(XSQLPageRequest, Element\)](#)

---

---

## Inherited Member Summary

---

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface [XSQLActionHandler](#)

[handleAction\(Node\)](#)

---

## Constructors

### XSQLActionHandlerImpl()

```
public XSQLActionHandlerImpl()
```

## Methods

### init(XSQLPageRequest, Element)

```
public void init(XSQLPageRequest env, oracle.xml.xsql.Element e)
```

#### **Specified By:**

[init\(XSQLPageRequest, Element\)](#) in interface [XSQLActionHandler](#)

## XSQLCommandLine

### Syntax

```
public final class XSQLCommandLine extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.xml.xsql.XSQLCommandLine
```

### Description

Command-line Utility to process XSQL Pages.

---

### Member Summary

---

Constructors

[XSQLCommandLine\(\)](#)

Methods

[main\(String\[\]\)](#)

---

---

### Inherited Member Summary

---

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

---

### Constructors

#### XSQLCommandLine()

```
public XSQLCommandLine()
```

### Methods

#### main(String[])

```
public static void main(java.lang.String[] args)
```

## XSQLDiagnostic

### Syntax

```
public final class XSQLDiagnostic extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLDiagnostic
```

### Description

---

#### Member Summary

---

##### Constructors

[XSQLDiagnostic\(String\)](#)

##### Methods

[debugPrintToFile\(String, String\)](#)

[msg\(String\)](#)

---

---

#### Inherited Member Summary

---

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

---

### Constructors

#### XSQLDiagnostic(String)

```
public XSQLDiagnostic(java.lang.String diagFile)
```

### Methods

#### debugPrintToFile(String, String)

```
public static void debugPrintToFile(java.lang.String msg, java.lang.String  
filename)
```



**msg(String)**

```
public void msg(java.lang.String text)
```

## XSQLHttpUtil

### Syntax

```
public final class XSQLHttpUtil extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLHttpUtil
```

### Description

---

#### Member Summary

---

##### Constructors

[XSQLHttpUtil\(\)](#)

##### Methods

[HttpRequestAsXMLDocument\(HttpServletRequest, String\)](#)

[XL\(String, String\)](#)

---

---

#### Inherited Member Summary

---

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

---

### Constructors

#### XSQLHttpUtil()

```
public XSQLHttpUtil()
```

### Methods

#### HttpRequestAsXMLDocument(HttpServletRequest, String)

```
public static oracle.xml.xsql.XMLDocument  
HttpRequestAsXMLDocument(oracle.xml.xsql.HttpServletRequest req,
```

```
java.lang.String XSQLPageEncoding)
```

**XL(String, String)**

```
public static java.lang.String XL(java.lang.String s, java.lang.String enc)
```

## XSQLPageRequest

### Syntax

```
public interface XSQLPageRequest
```

### All Known Implementing Classes:

[XSQLPageRequestImpl](#)

### Description

Interface representing a request for an XSQL Page

---

#### Member Summary

---

##### Methods

<a href="#">createNestedRequest(URL, Dictionary)</a>	Returns an instance of a nested Request
<a href="#">getConnectionName()</a>	Returns the name of the connection being used for this request May be null if no connection set/in-use.
<a href="#">getErrorWriter()</a>	Returns a PrintWriter to print out errors processing this request
<a href="#">getJDBCConnection()</a>	Gets the JDBC connection being used for this request (can be null)
<a href="#">getPageEncoding()</a>	Returns encoding of source XSQL Page associated with this request
<a href="#">getParameter(String)</a>	Returns the value of the requested parameter
<a href="#">getPostedDocument()</a>	Returns the content of Posted XML for this request as an XML Document
<a href="#">getRequestParamsAsXMLDocument()</a>	Returns the content of a Request parameters as an XML Document
<a href="#">getRequestType()</a>	Returns a string identifying the type of page request being made.
<a href="#">getSourceDocumentURI()</a>	Returns a String representation of the requested document's URI
<a href="#">getStylesheetParameter(String)</a>	Gets a stylesheet parameter by name
<a href="#">getStylesheetParameters()</a>	Gets an enumeration of stylesheet parameter names

---

**Member Summary**

---

<a href="#">getStylesheetURI()</a>	Returns the URI of the stylesheet to be used to process the result.
<a href="#">getUserAgent()</a>	Returns a String identifier of the requesting program
<a href="#">getWriter()</a>	Returns a PrintWriter used for writing out the results of a page request
<a href="#">getXSQLConnection()</a>	Gets the XSQLConnection Object being used for this request Might be null.
<a href="#">isIncludedRequest()</a>	Returns true if this request is being included in another.
<a href="#">isOracleDriver()</a>	Returns true if the current connection uses the Oracle JDBC Driver
<a href="#">printedErrorHandler()</a>	Returns the state of whether an Error Header has been printed
<a href="#">requestProcessed()</a>	Allows Page Request to Perform end-of-request processing
<a href="#">setConnectionName(String)</a>	Sets the connection name to use for this request
<a href="#">setContentTypes(String)</a>	Sets the content type of the resulting page
<a href="#">setIncludingRequest(XSQLPageRequest)</a>	Sets the Including Page Request object for this request.
<a href="#">setPageEncoding(String)</a>	Sets encoding of source XSQL page associated with this request.
<a href="#">setPageParam(String, String)</a>	Sets a dynamic page parameter value.
<a href="#">setPostedDocument(Document)</a>	Allows programmatic setting of the Posted Document
<a href="#">setPrintedErrorHandler(boolean)</a>	Sets whether an Error Header has been printed
<a href="#">setStylesheetParameter(String, String)</a>	Sets the value of a parameter to be passed to the associated stylesheet
<a href="#">setStylesheetURI(String)</a>	Sets the URI of the stylesheet to be used to process the result.
<a href="#">translateURL(String)</a>	Returns a string representing an absolute URL resolved relative to the base URI for this request.
<a href="#">useConnectionPooling()</a>	Returns true if connection pooling is desired for this request

---

**Member Summary**

---

[useHTMLErrors\(\)](#)Returns true if HTML-formatted error messages are desired for this request

---

**Methods****createNestedRequest(URL, Dictionary)**

```
public XSQLPageRequest createNestedRequest(java.net.URL pageurl,  
java.util.Dictionary params)
```

Returns an instance of a nested Request

**getConnectionName()**

```
public java.lang.String getConnectionName()
```

Returns the name of the connection being used for this request May be null if no connection set/in-use.

**getErrorWriter()**

```
public java.io.PrintWriter getErrorWriter()
```

Returns a PrintWriter to print out errors processing this request

**getJDBCConnection()**

```
public java.sql.Connection getJDBCConnection()
```

Gets the JDBC connection being used for this request (can be null)

**getPageEncoding()**

```
public java.lang.String getPageEncoding()
```

Returns encoding of source XSQL Page associated with this request

**getParameter(String)**

```
public java.lang.String getParameter(java.lang.String name)
```

Returns the value of the requested parameter

**Parameters:**name - the name of the parameter

**getPostedDocument()**

```
public oracle.xml.xsql.Document getPostedDocument()
```

Returns the content of Posted XML for this request as an XML Document

**getRequestParamsAsXMLDocument()**

```
public oracle.xml.xsql.Document getRequestParamsAsXMLDocument()
```

Returns the content of a Request parameters as an XML Document

**getRequestType()**

```
public java.lang.String getRequestType()
```

Returns a string identifying the type of page request being made.

**getSourceDocumentURI()**

```
public java.lang.String getSourceDocumentURI()
```

Returns a String representation of the requested document's URI

**getStylesheetParameter(String)**

```
public java.lang.String getStylesheetParameter(java.lang.String name)
```

Gets a stylesheet parameter by name

**getStylesheetParameters()**

```
public java.util.Enumeration getStylesheetParameters()
```

Gets an enumeration of stylesheet parameter names

**getStylesheetURI()**

```
public java.lang.String getStylesheetURI()
```

Returns the URI of the stylesheet to be used to process the result.

**getUserAgent()**

```
public java.lang.String getUserAgent()
```

Returns a String identifier of the requesting program

**getWriter()**

```
public java.io.PrintWriter getWriter()
```

Returns a PrintWriter used for writing out the results of a page request

### **getXSQLConnection()**

```
public oracle.xml.xsql.XSQLConnection getXSQLConnection()  
Gets the XSQLConnection Object being used for this request Might be null.
```

### **isIncludedRequest()**

```
public boolean isIncludedRequest()  
Returns true if this request is being included in another.
```

### **isOracleDriver()**

```
public boolean isOracleDriver()  
Returns true if the current connection uses the Oracle JDBC Driver
```

### **printedErrorHandler()**

```
public boolean printedErrorHandler()  
Returns the state of whether an Error Header has been printed
```

### **requestProcessed()**

```
public void requestProcessed()  
Allows Page Request to Perform end-of-request processing
```

### **setConnectionName(String)**

```
public void setConnectionName(java.lang.String connName)  
Sets the connection name to use for this request
```

### **setContentType(String)**

```
public void setContentType(java.lang.String mimetype)  
Sets the content type of the resulting page
```

### **setIncludingRequest(XSQLPageRequest)**

```
public void setIncludingRequest(XSQLPageRequest includingEnv)  
Sets the Including Page Request object for this request.
```

### **setPageEncoding(String)**

```
public void setPageEncoding(java.lang.String enc)  
Sets encoding of source XSQL page associated with this request.
```



**setPageParam(String, String)**

```
public void setPageParam(java.lang.String name, java.lang.String value)
```

Sets a dynamic page parameter value.

**setPostedDocument(Document)**

```
public void setPostedDocument(oracle.xml.xsql.Document doc)
```

Allows programmatic setting of the Posted Document

**setPrintedErrorHandler(boolean)**

```
public void setPrintedErrorHandler(boolean yes)
```

Sets whether an Error Header has been printed

**setStylesheetParameter(String, String)**

```
public void setStylesheetParameter(java.lang.String name, java.lang.String value)
```

Sets the value of a parameter to be passed to the associated stylesheet

**setStylesheetURI(String)**

```
public void setStylesheetURI(java.lang.String uri)
```

Sets the URI of the stylesheet to be used to process the result.

**translateURL(String)**

```
public java.lang.String translateURL(java.lang.String url)
```

Returns a string representing an absolute URL resolved relative to the base URI for this request.

**useConnectionPooling()**

```
public boolean useConnectionPooling()
```

Returns true if connection pooling is desired for this request

**useHTMLErrors()**

```
public boolean useHTMLErrors()
```

Returns true if HTML-formatted error messages are desired for this request

**setRequestObject**

```
void setRequestObject(java.lang.String name, java.lang.Object obj)
```

Sets a request-scope object

## **getRequestObject**

`java.lang.Object getRequestObject(java.lang.String name)`  
Gets a request-scope object

## XSQLPageRequestImpl

### Syntax

```
public abstract class XSQLPageRequestImpl extends java.lang.Object implements  
XSQLPageRequest
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLPageRequestImpl
```

### Direct Known Subclasses:

[XSQLServletPageRequest](#)

### All Implemented Interfaces:

[XSQLPageRequest](#)

## Description

Base implementation of the XSQLPageRequest interface that case be used to derive new kinds of page request implementations.

---

### Member Summary

---

#### Constructors

[XSQLPageRequestImpl\(\)](#)  
[XSQLPageRequestImpl\(Hashtable\)](#)  
[XSQLPageRequestImpl\(String, Hashtable\)](#)

#### Methods

<a href="#">getConnectionName()</a>	Returns the name of the connection being used for this request May be null if no connection set/in-use.
<a href="#">getErrorWriter()</a>	
<a href="#">getJDBCConnection()</a>	Gets the JDBC connection being used for this request (can be null)
<a href="#">getPageEncoding()</a>	Returns encoding of source XSQL Page associated with this request

---

**Member Summary**

---

<a href="#">getParameter(String)</a>	
<a href="#">getPostedDocument()</a>	
<a href="#">getRequestParamsAsXMLDocument()</a>	
<a href="#">getSourceDocumentURI()</a>	
<a href="#">getStylesheetParameter(String)</a>	Gets a stylesheet parameter by name
<a href="#">getStylesheetParameters()</a>	Gets an enumeration of stylesheet parameter names
<a href="#">getStylesheetURI()</a>	
<a href="#">getUserAgent()</a>	
<a href="#">getWriter()</a>	
<a href="#">getXSQLConnection()</a>	Gets the XSQLConnection Object being used for this request Might be null.
<a href="#">isIncludedRequest()</a>	Returns true if this request is being included in another.
<a href="#">isOracleDriver()</a>	Returns true if the current connection uses the Oracle JDBC Driver
<a href="#">printedErrorHeader()</a>	
<a href="#">requestProcessed()</a>	Allows Page Request to Perform end-of-request processing
<a href="#">setConnectionName(String)</a>	Sets the connection being used for this request (can be null)
<a href="#">setContentType(String)</a>	
<a href="#">setIncludingRequest(XSQLPageRequest)</a>	Sets the Including Page Request object for this request.
<a href="#">setPageEncoding(String)</a>	Associates an XSQL Page with the request
<a href="#">setPageParam(String, String)</a>	Sets a dynamic page parameter value.
<a href="#">setPostedDocument(Document)</a>	
<a href="#">setPrintedErrorHeader(boolean)</a>	
<a href="#">setStylesheetParameter(String, String)</a>	
<a href="#">setStylesheetURI(String)</a>	
<a href="#">translateURL(String)</a>	

---

**Member Summary**

---

[useConnectionPooling\(\)](#)[useHTMLErrors\(\)](#)

---

---

**Inherited Member Summary**

---

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface [XSQLPageRequest](#)

[createNestedRequest\(URL, Dictionary\)](#), [getRequestType\(\)](#)

---

**Constructors****XSQLPageRequestImpl()**

```
public XSQLPageRequestImpl()
```

**XSQLPageRequestImpl(Hashtable)**

```
public XSQLPageRequestImpl(java.util.Hashtable parameters)
```

**XSQLPageRequestImpl(String, Hashtable)**

```
public XSQLPageRequestImpl(java.lang.String pageurl, java.util.Hashtable parameters)
```

**Methods****getConnectionName()**

```
public java.lang.String getConnectionName()
```

Returns the name of the connection being used for this request May be null if no connection set/in-use.

**Specified By:**

[getConnectionName\(\)](#) in interface [XSQLPageRequest](#)

### **getErrorWriter()**

```
public java.io.PrintWriter getErrorWriter()
```

**Specified By:**

[getErrorWriter\(\)](#) in interface [XSQLPageRequest](#)

### **getJDBCConnection()**

```
public java.sql.Connection getJDBCConnection()
```

Gets the JDBC connection being used for this request (can be null)

**Specified By:**

[getJDBCConnection\(\)](#) in interface [XSQLPageRequest](#)

### **getPageEncoding()**

```
public java.lang.String getPageEncoding()
```

Returns encoding of source XSQL Page associated with this request

**Specified By:**

[getPageEncoding\(\)](#) in interface [XSQLPageRequest](#)

### **getParameter(String)**

```
public java.lang.String getParameter(java.lang.String name)
```

**Specified By:**

[getParameter\(String\)](#) in interface [XSQLPageRequest](#)

### **getPostedDocument()**

```
public oracle.xml.xsql.Document getPostedDocument()
```

**Specified By:**

[getPostedDocument\(\)](#) in interface [XSQLPageRequest](#)

### **getRequestParamsAsXMLDocument()**

```
public oracle.xml.xsql.Document getRequestParamsAsXMLDocument()
```

**Specified By:**

[getRequestParamsAsXMLDocument\(\)](#) in interface [XSQLPageRequest](#)

**getSourceDocumentURI()**

```
public java.lang.String getSourceDocumentURI()
```

**Specified By:**

[getSourceDocumentURI\(\)](#) in interface [XSQLPageRequest](#)

**getStylesheetParameter(String)**

```
public java.lang.String getStylesheetParameter(java.lang.String name)
```

Gets a stylesheet parameter by name

**Specified By:**

[getStylesheetParameter\(String\)](#) in interface [XSQLPageRequest](#)

**getStylesheetParameters()**

```
public java.util.Enumeration getStylesheetParameters()
```

Gets an enumeration of stylesheet parameter names

**Specified By:**

[getStylesheetParameters\(\)](#) in interface [XSQLPageRequest](#)

**getStylesheetURI()**

```
public java.lang.String getStylesheetURI()
```

**Specified By:**

[getStylesheetURI\(\)](#) in interface [XSQLPageRequest](#)

**getUserAgent()**

```
public java.lang.String getUserAgent()
```

**Specified By:**

[getUserAgent\(\)](#) in interface [XSQLPageRequest](#)

### getWriter()

```
public java.io.PrintWriter getWriter()
```

**Specified By:**

[getWriter\(\)](#) in interface [XSQLPageRequest](#)

### getXSQLConnection()

```
public oracle.xml.xsql.XSQLConnection getXSQLConnection()
```

Gets the XSQLConnection Object being used for this request Might be null.

**Specified By:**

[getXSQLConnection\(\)](#) in interface [XSQLPageRequest](#)

### isIncludedRequest()

```
public boolean isIncludedRequest()
```

Returns true if this request is being included in another.

**Specified By:**

[isIncludedRequest\(\)](#) in interface [XSQLPageRequest](#)

### isOracleDriver()

```
public boolean isOracleDriver()
```

Returns true if the current connection uses the Oracle JDBC Driver

**Specified By:**

[isOracleDriver\(\)](#) in interface [XSQLPageRequest](#)

### printedErrorHandler()

```
public boolean printedErrorHandler()
```

**Specified By:**

[printedErrorHandler\(\)](#) in interface [XSQLPageRequest](#)

### requestProcessed()

```
public void requestProcessed()
```

Allows Page Request to Perform end-of-request processing



**Specified By:**

[requestProcessed\(\)](#) in interface [XSQLPageRequest](#)

**setConnectionName(String)**

```
public void setConnectionName(java.lang.String connName)
```

Sets the connection being used for this request (can be null)

**Specified By:**

[setConnectionName\(String\)](#) in interface [XSQLPageRequest](#)

**setContentType(String)**

```
public void setContentType(java.lang.String mimetype)
```

**Specified By:**

[setContentType\(String\)](#) in interface [XSQLPageRequest](#)

**setIncludingRequest(XSQLPageRequest)**

```
public void setIncludingRequest(XSQLPageRequest includingEnv)
```

Sets the Including Page Request object for this request.

**Specified By:**

[setIncludingRequest\(XSQLPageRequest\)](#) in interface [XSQLPageRequest](#)

**setPageEncoding(String)**

```
public void setPageEncoding(java.lang.String enc)
```

Associates an XSQL Page with the request

**Specified By:**

[setPageEncoding\(String\)](#) in interface [XSQLPageRequest](#)

**setPageParam(String, String)**

```
public void setPageParam(java.lang.String name, java.lang.String value)
```

Sets a dynamic page parameter value.

**Specified By:**

[setPageParam\(String, String\)](#) in interface [XSQLPageRequest](#)

### setPostedDocument(Document)

```
public void setPostedDocument(oracle.xml.xsql.Document doc)
```

#### Specified By:

[setPostedDocument\(Document\)](#) in interface [XSQLPageRequest](#)

### setPrintedErrorHandler(boolean)

```
public void setPrintedErrorHandler(boolean yes)
```

#### Specified By:

[setPrintedErrorHandler\(boolean\)](#) in interface [XSQLPageRequest](#)

### setStylesheetParameter(String, String)

```
public void setStylesheetParameter(java.lang.String name, java.lang.String value)
```

#### Specified By:

[setStylesheetParameter\(String, String\)](#) in interface [XSQLPageRequest](#)

### setStylesheetURI(String)

```
public void setStylesheetURI(java.lang.String uri)
```

#### Specified By:

[setStylesheetURI\(String\)](#) in interface [XSQLPageRequest](#)

### translateURL(String)

```
public java.lang.String translateURL(java.lang.String filename)
```

#### Specified By:

[translateURL\(String\)](#) in interface [XSQLPageRequest](#)

### useConnectionPooling()

```
public boolean useConnectionPooling()
```

#### Specified By:

[useConnectionPooling\(\)](#) in interface [XSQLPageRequest](#)

**useHTMLErrors()**

```
public boolean useHTMLErrors()
```

**Specified By:**

[useHTMLErrors\(\)](#) in interface [XSQLPageRequest](#)

**setRequestObject**

```
void setRequestObject(java.lang.String name, java.lang.Object obj)
```

Sets a request-scope object

**getRequestObject**

```
java.lang.Object getRequestObject(java.lang.String name)
```

Gets a request-scope object

## XSQLParserHelper

### Syntax

```
public final class XSQLParserHelper extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLParserHelper
```

### Description

Common XML Parsing Routines

---

#### Member Summary

---

##### Constructors

[XSQLParserHelper\(\)](#)

##### Methods

[newDocument\(\)](#)

[parse\(InputStream, URL, PrintWriter\)](#)

[parse\(Reader, PrintWriter\)](#)

[parse\(URL, PrintWriter\)](#)

[parseFromString\(StringBuffer, PrintWriter\)](#)

[parseFromString\(String, PrintWriter\)](#)

[print\(Document, PrintWriter\)](#)

---

---

#### Inherited Member Summary

---

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

## Constructors

### XSQLParserHelper()

```
public XSQLParserHelper()
```

## Methods

### newDocument()

```
public static oracle.xml.xsql.Document newDocument()
```

### parse(InputStream, URL, PrintWriter)

```
public static oracle.xml.xsql.Document parse(java.io.InputStream is,  
java.net.URL baseUrl, java.io.PrintWriter errorWriter)
```

### parse(Reader, PrintWriter)

```
public static oracle.xml.xsql.Document parse(java.io.Reader r,  
java.io.PrintWriter errorWriter)
```

### parse(URL, PrintWriter)

```
public static oracle.xml.xsql.Document parse(java.net.URL url,  
java.io.PrintWriter errorWriter)
```

### parseFromString(StringBuffer, PrintWriter)

```
public static oracle.xml.xsql.Document parseFromString(java.lang.StringBuffer  
xmlString, java.io.PrintWriter errorWriter)
```

### parseFromString(String, PrintWriter)

```
public static oracle.xml.xsql.Document parseFromString(java.lang.String  
xmlString, java.io.PrintWriter errorWriter)
```

### print(Document, PrintWriter)

```
public static void print(oracle.xml.xsql.Document d, java.io.PrintWriter out)
```

## XSQLRequest

### Syntax

```
public class XSQLRequest extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLRequest
```

### Description

Programmatically process a request for an XSQL Page.

---

### Member Summary

---

#### Constructors

<a href="#">XSQLRequest(String)</a>	Create a Request for an XSQL Page
<a href="#">XSQLRequest(String, XSQLPageRequest)</a>	Create a Request for an XSQL Page
<a href="#">XSQLRequest(URL)</a>	Create a Request for an XSQL Page
<a href="#">XSQLRequest(oracle.xml.parser.v2.XMLDocument page, java.net.URL baseUrl)</a>	Create a Request for an XSQL Page
<a href="#">XSQLRequest(XSQLConnectionFactory fact, java.lang.String url)</a>	Create a Request for an XSQL Page using a custom connection manager factory
<a href="#">XSQLRequest(XSQLConnectionFactory fact, java.net.URL url)</a>	Create a Request for an XSQL Page with a custom connection manager factory.
<a href="#">XSQLRequest(XSQLConnectionFactory fact, oracle.xml.parser.v2.XMLDocument page, java.net.URL baseUrl)</a>	Create a Request for an XSQL Page
<a href="#">XSQLRequest(URL, XSQLPageRequest)</a>	Create a Request for an XSQL Page

#### Methods

---

**Member Summary**

---

<a href="#">process()</a>	Process the request, writing output/errors to System.out/System.err
<a href="#">process(Dictionary)</a>	Process the request, writing output/errors to System.out/System.err
<a href="#">process(Dictionary, PrintWriter, PrintWriter)</a>	Process the request, writing output/errors to respective PrintWriters
<a href="#">process(PrintWriter, PrintWriter)</a>	Process the request, writing output/errors to respective PrintWriters
<a href="#">processToXML()</a>	Process the request, writing output/errors to System.out/System.err
<a href="#">processToXML(Dictionary)</a>	Process the request, writing output/errors to System.out/System.err
<a href="#">processToXML(Dictionary, PrintWriter)</a>	Process the request, writing output/errors to respective PrintWriters
<a href="#">processToXML(PrintWriter)</a>	Process the request, writing errors to respective PrintWriters
<a href="#">setPostedDocument(Document)</a>	Programmatically set an XML Document to be treated the same as if it were posted as part of the request.

---

---

**Inherited Member Summary**

---

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

---

### Constructors

#### XSQLRequest(String)

```
public XSQLRequest(java.lang.String url)
```

Create a Request for an XSQL Page

**Parameters:**

url - String representation of an URL to an XSQL Page

#### XSQLRequest(String, XSQLPageRequest)

```
public XSQLRequest(java.lang.String url, XSQLPageRequest env)
```

Create a Request for an XSQL Page

**Parameters:**

url - String representation of an URL to an XSQL Page

env - Calling XSQLPageRequest environment

#### XSQLRequest(URL)

```
public XSQLRequest(java.net.URL url)
```

Create a Request for an XSQL Page

**Parameters:**

url - URL to an XSQL Page

#### XSQLRequest(URL, XSQLPageRequest)

```
public XSQLRequest(java.net.URL url, XSQLPageRequest env)
```

Create a Request for an XSQL Page

**Parameters:**

url - URL to an XSQL Page

env - Calling XSQLPageRequest environment

### Methods

#### process()

```
public void process()
```



Process the request, writing output/errors to System.out/System.err

### **process(Dictionary)**

```
public void process(java.util.Dictionary params)
```

Process the request, writing output/errors to System.out/System.err

#### **Parameters:**

params - Dictionary (e.g. Hashtable) with XSQL Page parameters

### **process(Dictionary, PrintWriter, PrintWriter)**

```
public void process(java.util.Dictionary params, java.io.PrintWriter out,  
java.io.PrintWriter err)
```

Process the request, writing output/errors to respective PrintWriters

#### **Parameters:**

params - Dictionary (e.g. Hashtable) with XSQL Page parameters

out - PrintWriter to use to write the resulting page results

err - PrintWriter to use to write the resulting page errors

### **process(PrintWriter, PrintWriter)**

```
public void process(java.io.PrintWriter out, java.io.PrintWriter err)
```

Process the request, writing output/errors to respective PrintWriters

#### **Parameters:**

out - PrintWriter to use to write the resulting page results

err - PrintWriter to use to write the resulting page errors

### **processToXML()**

```
public org.w3c.dom.Document processToXML()
```

Process the request, writing output/errors to System.out/System.err

### **processToXML(Dictionary)**

```
public org.w3c.dom.Document processToXML(java.util.Dictionary params)
```

Process the request, writing output/errors to System.out/System.err

**Parameters:**

params - Dictionary (e.g. Hashtable) with XSQL Page parameters

**processToXML(Dictionary, PrintWriter)**

```
public org.w3c.dom.Document processToXML(java.util.Dictionary params,  
java.io.PrintWriter err)
```

Process the request, writing output/errors to respective PrintWriters

**Parameters:**

params - Dictionary (e.g. Hashtable) with XSQL Page parameters

err - PrintWriter to use to write the resulting page errors

**processToXML(PrintWriter)**

```
public org.w3c.dom.Document processToXML(java.io.PrintWriter err)
```

Process the request, writing errors to respective PrintWriters

**Parameters:**

err - PrintWriter to use to write the resulting page errors

**setPostedDocument(Document)**

```
public void setPostedDocument(org.w3c.dom.Document postedDoc)
```

Programmatically set an XML Document to be treated the same as if it were posted as part of the request.

**Parameters:**

postedDoc - DOM Document

**XSQLRequestObjectListener**

Interface that an object created by an action handler can implement to be notified when the current page request processing is completed. It has a single method:

```
void pageProcessingCompleted()
```

Objects that implement this interface and which are added to the current request context using `XSQLPageRequest::setRequestObject()` will be notified when the page processing of the outermost page is completed.

## XSQLServlet

### Syntax

```
public final class XSQLServlet  
  
oracle.xml.xsql.XSQLServlet
```

### Description

Servlet to enable HTTP GET-ing of and POST-ing to XSQL Pages

---

### Member Summary

---

#### Constructors

[XSQLServlet\(\)](#)

#### Methods

[doGet\(HttpServletRequest, HttpServletResponse\)](#)

[doPost\(HttpServletRequest, HttpServletResponse\)](#)

[getServletInfo\(\)](#)

[init\(ServletConfig\)](#)

[inJServ\(\)](#)

---

### Constructors

#### XSQLServlet()

```
public XSQLServlet()
```

### Methods

#### doGet(HttpServletRequest, HttpServletResponse)

```
public void doGet(oracle.xml.xsql.HttpServletRequest request,  
oracle.xml.xsql.HttpServletResponse response)
```

**doPost(HttpServletRequest, HttpServletResponse)**

```
public void doPost(oracle.xml.xsql.HttpServletRequest request,  
oracle.xml.xsql.HttpServletResponse response)
```

**getServletInfo()**

```
public java.lang.String getServletInfo()
```

**init(ServletConfig)**

```
public void init(oracle.xml.xsql.ServletConfig config)
```

**inJServ()**

```
public static boolean inJServ()
```

# XSQLServletPageRequest

## Syntax

```
public final class XSQLServletPageRequest extends XSQLPageRequestImpl
```

```
java.lang.Object
|
+--XSQLPageRequestImpl
|
+--oracle.xml.xsql.XSQLServletPageRequest
```

## All Implemented Interfaces:

[XSQLPageRequest](#)

## Description

Implementation of XSQLPageRequest for Servlet-based XSQL Page requests.

---

## Member Summary

---

### Constructors

[XSQLServletPageRequest\(HttpServletRequest, HttpServletResponse\)](#)

### Methods

<a href="#">createNestedRequest(URL, Dictionary)</a>	Returns an instance of a nested Request
<a href="#">getCookie(String)</a>	
<a href="#">getHttpServletRequest()</a>	Get the HttpServletRequest that initiated this XSQL Page Request.
<a href="#">getHttpServletResponse()</a>	Get the HttpServletResponse that is associated with this XSQL Page Request
<a href="#">getParameter(String)</a>	Use HTTP Parameters as the source of parameters instead
<a href="#">getPostedDocument()</a>	
<a href="#">getRequestParamsAsXMLDocument()</a>	

---

### Member Summary

---

[getRequestType\(\)](#)  
[getUserAgent\(\)](#)  
[setContentType\(String\)](#)  
[setPageEncoding\(String\)](#)  
[translateURL\(String\)](#)  
[useHTMLErrors\(\)](#)

---

---

### Inherited Member Summary

---

#### Methods inherited from class *XSQLPageRequestImpl*

[getConnectionName\(\)](#), [getErrorWriter\(\)](#), [getJDBCConnection\(\)](#), [getPageEncoding\(\)](#), [getSourceDocumentURI\(\)](#), [getStylesheetParameter\(String\)](#), [getStylesheetParameters\(\)](#), [getStylesheetURI\(\)](#), [getWriter\(\)](#), [getXSQLConnection\(\)](#), [isIncludedRequest\(\)](#), [isOracleDriver\(\)](#), [printedErrorHandler\(\)](#), [requestProcessed\(\)](#), [setConnectionName\(String\)](#), [setIncludingRequest\(XSQLPageRequest\)](#), [setPageParam\(String, String\)](#), [setPostedDocument\(Document\)](#), [setPrintedErrorHandler\(boolean\)](#), [setStylesheetParameter\(String, String\)](#), [setStylesheetURI\(String\)](#), [useConnectionPooling\(\)](#)

#### Methods inherited from class *java.lang.Object*

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

#### Methods inherited from interface *XSQLPageRequest*

[getConnectionName\(\)](#), [getErrorWriter\(\)](#), [getJDBCConnection\(\)](#), [getPageEncoding\(\)](#), [getSourceDocumentURI\(\)](#), [getStylesheetParameter\(String\)](#), [getStylesheetParameters\(\)](#), [getStylesheetURI\(\)](#), [getWriter\(\)](#), [getXSQLConnection\(\)](#), [isIncludedRequest\(\)](#), [isOracleDriver\(\)](#), [printedErrorHandler\(\)](#), [requestProcessed\(\)](#), [setConnectionName\(String\)](#), [setIncludingRequest\(XSQLPageRequest\)](#), [setPageParam\(String, String\)](#), [setPostedDocument\(Document\)](#), [setPrintedErrorHandler\(boolean\)](#), [setStylesheetParameter\(String, String\)](#), [setStylesheetURI\(String\)](#), [useConnectionPooling\(\)](#)

---

## Constructors

### XSQLServletPageRequest(HttpServletRequest, HttpServletResponse)

```
public XSQLServletPageRequest(oracle.xml.xsql.HttpServletRequest req,
oracle.xml.xsql.HttpServletResponse resp)
```



## Methods

### createNestedRequest(URL, Dictionary)

```
public XSQLPageRequest createNestedRequest(java.net.URL pageurl,  
java.util.Dictionary params)
```

Returns an instance of a nested Request

### getCookie(String)

```
public java.lang.String getCookie(java.lang.String name)
```

### getHttpServletRequest()

```
public oracle.xml.xsql.HttpServletRequest getHttpServletRequest()
```

Get the HttpServletRequest that initiated this XSQL Page Request.

### getHttpServletResponse()

```
public oracle.xml.xsql.HttpServletResponse getHttpServletResponse()
```

Get the HttpServletResponse that is associated with this XSQL Page Request

### getParameter(String)

```
public java.lang.String getParameter(java.lang.String name)
```

Use HTTP Parameters as the source of parameters instead

#### Overrides:

[getParameter\(String\)](#) in class [XSQLPageRequestImpl](#)

### getPostedDocument()

```
public oracle.xml.xsql.Document getPostedDocument()
```

#### Overrides:

[getPostedDocument\(\)](#) in class [XSQLPageRequestImpl](#)

### getRequestParamsAsXMLDocument()

```
public oracle.xml.xsql.Document getRequestParamsAsXMLDocument()
```

#### Overrides:

[getRequestParamsAsXMLDocument\(\)](#) in class [XSQLPageRequestImpl](#)

### **getRequestType()**

```
public java.lang.String getRequestType()
```

### **getUserAgent()**

```
public java.lang.String getUserAgent()
```

#### **Overrides:**

[getUserAgent\(\)](#) in class [XSQLPageRequestImpl](#)

### **setContentType(String)**

```
public void setContentType(java.lang.String mimetype)
```

#### **Overrides:**

[setContentType\(String\)](#) in class [XSQLPageRequestImpl](#)

### **setPageEncoding(String)**

```
public void setPageEncoding(java.lang.String enc)
```

#### **Overrides:**

[setPageEncoding\(String\)](#) in class [XSQLPageRequestImpl](#)

### **translateURL(String)**

```
public java.lang.String translateURL(java.lang.String path)
```

#### **Overrides:**

[translateURL\(String\)](#) in class [XSQLPageRequestImpl](#)

### **useHTMLErrors()**

```
public boolean useHTMLErrors()
```

#### **Overrides:**

[useHTMLErrors\(\)](#) in class [XSQLPageRequestImpl](#)

---

# XSQLStyleSheetProcessor

## Syntax

```
public final class XSQLStyleSheetProcessor extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.xml.xsql.XSQLStyleSheetProcessor
```

## Description

XSLT Stylesheet Processing Engine

---

### Member Summary

---

Constructors

[XSQLStyleSheetProcessor\(\)](#)

Methods

[processToDocument\(Document, String, XSQLPageRequest\)](#)

[processToWriter\(Document, String, XSQLPageRequest\)](#)

---

---

### Inherited Member Summary

---

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

---

## Constructors

### XSQLStyleSheetProcessor()

```
public XSQLStyleSheetProcessor()
```

## Methods

### **processToDocument(Document, String, XSQLPageRequest)**

```
public static oracle.xml.xsql.Document  
processToDocument(oracle.xml.xsql.Document xml, java.lang.String xslURI,  
XSQLPageRequest env)
```

### **processToWriter(Document, String, XSQLPageRequest)**

```
public static void processToWriter(oracle.xml.xsql.Document xml,  
java.lang.String xslURI, XSQLPageRequest env)
```

### **getServletContext()**

```
javax.servlet.ServletContext getServletContext()
```

Gets the Http Servlet Context

# XSQLUtil

## Syntax

```
public final class XSQLUtil extends java.lang.Object
```

```
java.lang.Object  
|  
+--oracle.xml.xsql.XSQLUtil
```

## Description

---

### Member Summary

---

#### Constructors

[XSQLUtil\(\)](#)

#### Methods

[DictionaryOfParamsAsXMLDocument\(Dictionary\)](#)

[safeURLAsString\(URL\)](#)

[select\(Document, String\)](#)

[select\(Element, String\)](#)

[select\(XMLDocument, String\)](#)

[select\(XMLElement, String\)](#)

[selectFirst\(Document, String\)](#)

[selectFirst\(Element, String\)](#)

[selectFirst\(XMLDocument, String\)](#)

[selectFirst\(XMLElement, String\)](#)

[stringParamValue\(Object\)](#)

[translate\(String, String\)](#)

[translate\(URL, String\)](#)

[valueOf\(Element, String\)](#)

[valueOf\(Node, String\)](#)

[valueOf\(XMLElement, String\)](#)

---

**Member Summary**

---

[valueOf\(XMLNode, String\)](#)[XL\(String, String\)](#)

---

---

**Inherited Member Summary**

---

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

**Constructors****XSQLUtil()**

```
public XSQLUtil()
```

**Methods****DictionaryOfParamsAsXMLDocument(Dictionary)**

```
public static oracle.xml.xsql.XMLDocument  
DictionaryOfParamsAsXMLDocument(java.util.Dictionary dict)
```

**safeURLAsString(URL)**

```
public static java.lang.String safeURLAsString(java.net.URL u)
```

**select(Document, String)**

```
public static oracle.xml.xsql.NodeList select(oracle.xml.xsql.Document d,  
java.lang.String pattern)
```

**select(Element, String)**

```
public static oracle.xml.xsql.NodeList select(oracle.xml.xsql.Element n,  
java.lang.String pattern)
```

**select(XMLDocument, String)**

```
public static oracle.xml.xsql.NodeList select(oracle.xml.xsql.XMLDocument d,  
java.lang.String pattern)
```

**select(XMLElement, String)**

```
public static oracle.xml.xsql.NodeList select(oracle.xml.xsql.XMLElement n,  
java.lang.String pattern)
```

**selectFirst(Document, String)**

```
public static oracle.xml.xsql.Node selectFirst(oracle.xml.xsql.Document d,  
java.lang.String pattern)
```

**selectFirst(Element, String)**

```
public static oracle.xml.xsql.Node selectFirst(oracle.xml.xsql.Element n,  
java.lang.String pattern)
```

**selectFirst(XMLDocument, String)**

```
public static oracle.xml.xsql.Node selectFirst(oracle.xml.xsql.XMLDocument d,  
java.lang.String pattern)
```

**selectFirst(XMLElement, String)**

```
public static oracle.xml.xsql.Node selectFirst(oracle.xml.xsql.XMLElement n,  
java.lang.String pattern)
```

**stringParamValue(Object)**

```
public static java.lang.String stringParamValue(java.lang.Object val)
```

**translate(String, String)**

```
public static java.lang.String translate(java.lang.String u, java.lang.String  
path)
```

**translate(URL, String)**

```
public static java.lang.String translate(java.net.URL u, java.lang.String path)
```

**valueOf(Element, String)**

```
public static java.lang.String valueOf(oracle.xml.xsql.Element n,  
java.lang.String pattern)
```

**valueOf(Node, String)**

```
public static java.lang.String valueOf(oracle.xml.xsql.Node n, java.lang.String  
pattern)
```

### **valueOf(XMLElement, String)**

```
public static java.lang.String valueOf(oracle.xml.xsql.XMLElement n,  
java.lang.String pattern)
```

### **valueOf(XMLNode, String)**

```
public static java.lang.String valueOf(oracle.xml.xsql.XMLNode n,  
java.lang.String pattern)
```

### **XL(String, String)**

```
public static java.lang.String XL(java.lang.String s, java.lang.String enc)
```

## **XSQLRequestObjectListener**

Interface that an object created by an action handler can implement to be notified when the current page request processing is completed.

It has a single method:

```
void pageProcessingCompleted()
```

Objects that implement this interface and which are added to the current request context using `XSQLPageRequest::setRequestObject()` will be notified when the page processing of the outermost page is



completed.

The following constructors were added to oracle.xml.xsql.XSQLRequest class:

XSQLRequest(oracle.xml.parser.v2.XMLDocument page, java.net.URL baseUrl)  
Create a Request for an XSQL Page

XSQLRequest(XSQLConnectionFactory fact, java.lang.String url)  
Create a Request for an XSQL Page using a custom connection manager factory

XSQLRequest(XSQLConnectionFactory fact, java.net.URL url)  
Create a Request for an XSQL Page with a custom connection manager factory.

XSQLRequest(XSQLConnectionFactory fact, oracle.xml.parser.v2.XMLDocument page, java.net.URL baseUrl)  
Create a Request for an XSQL Page

=====  
The following are new interfaces  
=====

-----  
public interface XSQLConnectionFactory

One of two interfaces that must be implemented to override the built-in connection manager implementation. The XSQL Page Processor asks the XSQLConnectionFactory associated with each request to create() an instance of an XSQLConnectionFactory to service the current request.

In multithreaded environments, the implementation of XSQLConnectionFactory must insure that an XSQLConnection instance returned by getConnection() is not used by another thread until it has been released by the XSQL Page Processor after the call to releaseConnection().

### Method Summary

XSQLConnection getConnection(java.lang.String connName, XSQLPageRequest env)

```
void releaseConnection(XSQLConnection theConn, XSQLPageRequest env)
```

-----  
P  
public interface XSQLConnectionFactory

One of two interfaces that must be implemented to override the built-in connection manager implementation. The XSQL Page Processor asks the XSQLConnectionFactory associated with each request to create() an instance of an XSQLConnectionFactory to service the current request.

#### Method Summary

```
XSQLConnectionFactory create()
```

-----

```
public interface XSQLDocumentSerializer
```

Interface that must be implemented by all XSQL Serializers which serialize an XSQL data page as an XML Document to a PrintWriter.

Upon encountering a serializer="XXX" pseudo-attribute in an <?xml-stylesheet?> processing instruction, the XSQL Page Processor invokes the associated serializer by:

- Constructing an instance of the serializer using the no-args constructor
- Invoking the XSQL document serializer's serialize() method

NOTE: An implementation of XSQLDocumentSerializer is expected to do the following actions.

- First, call env.setContentType() to set the content type

- Then, call `env.getWriter()` to get the `Writer` to write to

If the serializer throws an unhandled exception, the XSQL Page Processor will format the stacktrace.

See `oracle.xml.xsql.src.serializers.XSQLSampleSerializer` for an example.

### Method Summary

`void serialize(org.w3c.dom.Document doc, XSQLPageRequest env)`



---

---

# Transviewer Beans

## Package oracle.xml.async

---

### Class Summary

---

#### Interfaces

<a href="#">DOMBuilderErrorListener</a>	This interface must be implemented in order to receive notifications when error is found during parsing.
<a href="#">DOMBuilderListener</a>	This interface must be implemented in order to receive notifications about events during the asynchronous parsing.
<a href="#">XSLTransformerErrorListener</a>	This interface must be implemented in order to receive notifications about error events during the asynchronous transformation.
<a href="#">XSLTransformerListener</a>	

---

## DOMBuilder

### Syntax

```
public class DOMBuilder extends java.lang.Object implements  
java.io.Serializable, oracle.xml.async.DOMBuilderConstants, java.lang.Runnable
```

```
java.lang.Object  
|  
+--oracle.xml.async.DOMBuilder
```

### All Implemented Interfaces:

```
oracle.xml.async.DOMBuilderConstants, java.lang.Runnable, java.io.Serializable
```

### Description

This class encapsulates an eXtensible Markup Language (XML) 1.0 parser to parse an XML document and build a DOM tree. The parsing is done in a separate thread and DOMBuilderInterface must be used for notification when the tree is built.

### Fields

#### inSource

```
protected org.xml.sax.InputSource inSource  
InputSource containing XML data to parse
```

#### inStream

```
protected java.io.InputStream inStream  
InputStream containing XML data to parse
```

#### inString

```
protected java.lang.String inString  
String containing the URL to parse XML data from
```

### methodToCall

```
protected int methodToCall  
XML Parser method to call based on input types
```

## reader

protected java.io.Reader reader  
java.io.Reader containing XML data to be parsed

## result

protected oracle.xml.async.XMLDocument result  
XML Document being parsed

## rootName

protected java.lang.String rootName  
Name of the XML element to be treated as root

## url

protected java.net.URL url  
URL to parse XML data from

## Constructors

### DOMBuilder()

public DOMBuilder()  
Creates a new parser object.

### DOMBuilder(int)

public DOMBuilder(int id)  
Creates a new parser object with a given id.

#### Parameters:

[id](#) - The [DOMBuilder](#) id.

## Methods

### addDOMBuilderErrorListener(DOMBuilderErrorListener)

public void addDOMBuilderErrorListener([DOMBuilderErrorListener](#) p0)  
Adds DOMBuilderErrorListener



**Parameters:**

[p1](#) - The [DOMBuilderErrorListener](#) to add

**addDOMBuilderListener(DOMBuilderListener)**

```
public void addDOMBuilderListener(DOMBuilderListener p0)
```

Adds DOMBuilderListener

**Parameters:**

[p1](#) - The [DOMBuilderListener](#) to add

**getDoctype()**

```
public synchronized oracle.xml.async.DTD getDoctype()
```

Get the DTD

**Returns:**

The [DTD](#)

**getDocument()**

```
public synchronized oracle.xml.async.XMLDocument getDocument()
```

Gets the document

**Returns:**

The document being parsed

**getId()**

```
public int getId()
```

Returns the parser object id.

**Returns:**

The [DOMBuilder](#) id

**getReleaseVersion()**

```
public synchronized java.lang.String getReleaseVersion()
```

Returns the release version of the Oracle XML Parser

**Returns:**

the release version string

**getResult()**

```
public synchronized org.w3c.dom.Document getResult()  
Gets the document
```

**Returns:**

The document being parsed

**getValidationMode()**

```
public synchronized boolean getValidationMode()  
Returns the validation mode
```

**Returns:**

[true](#) if the XML parser is validating [false](#) if not

**parse(InputSource)**

```
public final synchronized void parse(org.xml.sax.InputSource in)  
Parses the XML from given input source
```

**Parameters:**

[in](#) - the [org.xml.sax.InputSource](#) to parse

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**parse(InputStream)**

```
public final synchronized void parse(java.io.InputStream in)  
Parses the XML from given input stream. The base URL should be set for resolving  
external entities and DTD.
```

**Parameters:**

[in](#) - the [InputStream](#) containing XML data to parse.

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**See Also:**

`oracle.xml.parser.v2.XMLParser`

**parse(Reader)**

```
public final synchronized void parse(java.io.Reader r)
```

Parses the XML from given input stream. The base URL should be set for resolving external entities and DTD.

**Parameters:**

`r` - the [Reader](#) containing XML data to parse.

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**See Also:**

`oracle.xml.parser.v2.XMLParser`

**parse(String)**

```
public final synchronized void parse(java.lang.String in)
```

Parses the XML from the URL indicated

**Parameters:**

`in` - the [String](#) containing the URL to parse from

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

## parse(URL)

```
public final synchronized void parse(java.net.URL url)
```

Parses the XML document pointed to by the given URL and creates the corresponding XML document hierarchy.

### Parameters:

[url](#) - the url points to the XML document to parse.

### Throws:

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

## parseDTD(InputSource, String)

```
public final synchronized void parseDTD(org.xml.sax.InputSource in,  
java.lang.String rootName)
```

Parses the XML External DTD from given input source

### Parameters:

[in](#) - the [org.xml.sax.InputSource](#) to parse

[rootName](#) - the element to be used as root Element

### Throws:

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

## parseDTD(InputStream, String)

```
public final synchronized void parseDTD(java.io.InputStream in, java.lang.String  
rootName)
```

Parses the XML External DTD from given input stream. The base URL should be set for resolving external entities and DTD.

**Parameters:**

[in](#) - the [InputStream](#) containing XML data to parse.

[rootName](#) - the element to be used as root Element

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**See Also:**

`oracle.xml.parser.v2.XMLParser`

**parseDTD(Reader, String)**

```
public final synchronized void parseDTD(java.io.Reader r, java.lang.String  
rootName)
```

Parses the XML External DTD from given input stream. The base URL should be set for resolving external entities and DTD.

**Parameters:**

[r](#) - the [Reader](#) containing XML data to parse.

[rootName](#) - the element to be used as root Element

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**See Also:**

`oracle.xml.parser.v2.XMLParser`

**parseDTD(String, String)**

```
public final synchronized void parseDTD(java.lang.String in, java.lang.String  
rootName)
```

Parses the XML External DTD from the URL indicated

**Parameters:**

[in](#) - the [String](#) containing the URL to parse from

[rootName](#) - the element to be used as root Element

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**parseDTD(URL, String)**

```
public final synchronized void parseDTD(java.net.URL url, java.lang.String
rootName)
```

Parses the XML External DTD document pointed to by the given URL and creates the corresponding XML document hierarchy.

**Parameters:**

[url](#) - the url points to the XML document to parse.

[rootName](#) - the element to be used as root Element

**Throws:**

[XMLParseException](#) - if syntax or other error encountered.

[SAXException](#) - Any SAX exception, possibly wrapping another exception.

[IOException](#) - IO Error.

**removeDOMBuilderErrorListener(DOMBuilderErrorListener)**

```
public synchronized void removeDOMBuilderErrorListener(DOMBuilderErrorListener
p0)
```

Remove DOMBuilderErrorListener

**Parameters:**

[p1](#) - The [DOMBuilderErrorListener](#) to remove

**removeDOMBuilderListener(DOMBuilderListener)**

```
public synchronized void removeDOMBuilderListener(DOMBuilderListener p0)
```

Remove DOMBuilderListener

**Parameters:**

[p1](#) - The [DOMBuilderListener](#) to remove

**run()**

```
public void run()  
This method runs in a thread
```

**Specified By:**

java.lang.Runnable.run() in interface java.lang.Runnable

**setBaseURL(URL)**

```
public synchronized void setBaseURL(java.net.URL url)  
Set the base URL for loading external entities and DTDs. This method should be  
called if the parse(InputStream) is used to parse the XML Document
```

**Parameters:**

[url](#) - The base URL

**setDebugMode(boolean)**

```
public void setDebugMode(boolean flag)  
Sets a flag to turn on debug information in the document
```

**Parameters:**

[flag](#) - determines whether debug info is stored

**setDoctype(DTD)**

```
public synchronized void setDoctype(oracle.xml.async.DTD dtd)  
Set the DTD
```

**Parameters:**

[dtd](#) - [DTD](#) to set and used while parsing

**setErrorStream(OutputStream)**

```
public final synchronized void setErrorStream(java.io.OutputStream out)  
Creates an output stream for the output of errors and warnings. If an output stream  
for errors is not specified, the parser will use the standard error output stream  
System.err for outputting errors and warnings.
```

**Parameters:**

[out](#) - The output stream to use for errors and warnings

**setErrorStream(OutputStream, String)**

```
public final synchronized void setErrorStream(java.io.OutputStream out,  
java.lang.String enc)
```

Creates an output stream for the output of errors and warnings. If an output stream for errors is not specified, the parser will use the standard error output stream [System.err](#) for outputting errors and warnings. Additionally, an `.exception` is thrown if the encoding specified is unsupported.

**Parameters:**

[out](#) - The output stream to use for errors and warnings

[enc](#) - the encoding to use

**Throws:**

[IOException](#) - if an unsupported encoding is specified

**setErrorStream(PrintWriter)**

```
public final synchronized void setErrorStream(java.io.PrintWriter out)
```

Creates an output stream for the output of errors and warnings. If an output stream for errors is not specified, the parser will use the standard error output stream [System.err](#) for outputting errors and warnings.

**Parameters:**

[out](#) - The [PrintWriter](#) to use for errors and warnings

**setNodeFactory(NodeFactory)**

```
public synchronized void setNodeFactory(oracle.xml.async.NodeFactory factory)
```

Set the node factory. Applications can extend the `NodeFactory` and register it through this method. The parser will then use the user supplied `NodeFactory` to create nodes of the DOM tree.

**Parameters:**

[factory](#) - The [NodeFactory](#) to set



**Throws:**

[XMLParseException](#) - if an invalid factory is set

**See Also:**

NodeFactory

**setPreserveWhitespace(boolean)**

```
public synchronized void setPreserveWhitespace(boolean flag)
```

Set the white space preserving mode

**Parameters:**

[flag](#) - preserving mode

**setValidationMode(boolean)**

```
public synchronized void setValidationMode(boolean yes)
```

Set the validation mode

**Parameters:**

[yes](#) - determines whether the XML parser should be validating

**showWarnings(boolean)**

```
public synchronized void showWarnings(boolean yes)
```

Switch to determine whether to print warnings

**Parameters:**

[yes](#) - determines whether warnings should be shown

## DOMBuilderBeanInfo

### Syntax

```
public class DOMBuilderBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
|
+--java.beans.SimpleBeanInfo
|
+--oracle.xml.async.DOMBuilderBeanInfo
```

### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Description

This class provides information about the DOMBuilder Bean.

## Constructors

### DOMBuilderBeanInfo()

```
public DOMBuilderBeanInfo()
```

The default Constructor

## Methods

### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

Gets an image object that can be used to represent [DOMBuilder](#) bean in toolbars, toolboxes, etc.

### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

### Parameters:

[iconKind](#) - The kind of icon requested.

**Returns:**

An image object representing the requested icon type for [DOMBuilder](#) bean.

**getPropertyDescriptors()**

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

Gets the [DOMBuilder](#) bean's [PropertyDescriptors](#)

**Overrides:**

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class  
java.beans.SimpleBeanInfo

**Returns:**

An array of PropertyDescriptors describing the editable properties supported by [DOMBuilder](#) bean.

## DOMBuilderErrorEvent

### Syntax

```
public class DOMBuilderErrorEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--oracle.xml.async.DOMBuilderErrorEvent
```

### All Implemented Interfaces:

```
java.io.Serializable
```

### Description

This class defines the error event which is sent when parse exception occurs.

### Fields

```
protected java.lang.Exception e  
The exception being raised.
```

### Constructors

#### DOMBuilderErrorEvent(Object, Exception)

```
public DOMBuilderErrorEvent(java.lang.Object p0, java.lang.Exception e)  
Constructor for DOMBuilderErrorEvent.
```

#### Parameters:

[p0](#) - The [Object](#) that created this event.

[e](#) - The [Exception](#) raised.

### Methods

#### getException()

```
public java.lang.Exception getException()  
Gets the Exception
```

**Returns:**

The Exception beind raised

**getMessage()**

```
public java.lang.String getMessage()
```

Returns the error message generated by the parser

**Returns:**

The error message string

## DOMBuilderErrorListener

### Syntax

```
public interface DOMBuilderErrorListener extends java.util.EventListener
```

### All Superinterfaces:

```
java.util.EventListener
```

### Description

This interface must be implemented in order to receive notifications when error is found during parsing. The class implementing this interface must be added to the DOMBuilder using addDOMBuilderErrorListener method.

## Methods

### domBuilderErrorCalled(DOMBuilderErrorEvent)

```
public void domBuilderErrorCalled(DOMBuilderErrorEvent p0)
```

This method is called when a parse error occurs.

### Parameters:

[p0](#) - The DOMBuilderErrorEvent object produced by the DOMBuilder as result of parsing error

## DOMBuilderEvent

### Syntax

```
public class DOMBuilderEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--oracle.xml.async.DOMBuilderEvent
```

### All Implemented Interfaces:

```
java.io.Serializable
```

### Description

The event object that DOMBuilder uses to notify all registered listeners about parse events.

### Fields

#### id

```
protected int id
ID of the source DOMBuilder object
```

### Constructors

#### DOMBuilderEvent(Object, int)

```
public DOMBuilderEvent(java.lang.Object p0, int p1)
Creates a new DOMBuilderEvent
```

#### Parameters:

[p0](#) - The [Object](#) creating this event.

[p1](#) - Id of the [DOMBuilder](#) creating this event.

## Methods

### getID()

```
public int getID()
```

Returns unique id of the DOMBuilder object which can be used to identify which instance of the DOMBuilder generated this event in cases where multiple instances of DOMBuilder may be working in background.

**Returns:**

The unique [id](#) of the source DOMBuilder for this event.



## DOMBuilderListener

### Syntax

```
public interface DOMBuilderListener extends java.util.EventListener
```

### All Superinterfaces:

`java.util.EventListener`

### Description

This interface must be implemented in order to receive notifications about events during the asynchronous parsing. The class implementing this interface must be added to the DOMBuilder using `addDOMBuilderListener` method.

## Methods

### `domBuilderError(DOMBuilderEvent)`

```
public void domBuilderError(DOMBuilderEvent p0)
```

This method is called when parse error occur.

#### Parameters:

[p0](#) -- The DOMBuilderEvent object produced by the DOMBuilder

### `domBuilderOver(DOMBuilderEvent)`

```
public void domBuilderOver(DOMBuilderEvent p0)
```

This method is called when the parse is complete

#### Parameters:

[p0](#) -- The DOMBuilderEvent object produced by the DOMBuilder

### `domBuilderStarted(DOMBuilderEvent)`

```
public void domBuilderStarted(DOMBuilderEvent p0)
```

This method is called when parse starts

**Parameters:**

[p0](#) - - The DOMBuilderEvent object produced by the DOMBuilder

# ResourceManager

## Syntax

```
public class ResourceManager extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.xml.async.ResourceManager
```

## Constructors

### ResourceManager(int)

```
public ResourceManager(int i)
```

The ResourceManager constructor

#### Parameters:

`<code>i</code>` - - the number of resources to manage

## Methods

### activeFound()

```
public boolean activeFound()
```

Checks if any of the logical resources being managed are in active use

#### Returns:

[true](#) - if one or more resource is in use [false](#) - if none of the resources are in use

### getResource()

```
public synchronized void getResource()
```

If the number of resources available for use is nonzero, the method decreases the number of resources by one. Otherwise, it waits until a resource is released & it becomes available for use.

### releaseResource()

```
public void releaseResource()
```

Releases a resource. When this method is called, the number of resources available is increased by one.

## **sleep(int)**

```
public void sleep(int i)
```

Allows usage of Thread.sleep() without try/catch

# XSLTransformer

## Syntax

```
public class XSLTransformer extends java.lang.Object implements  
java.io.Serializable, oracle.xml.async.XSLTransformerConstants,  
java.lang.Runnable
```

```
java.lang.Object  
|  
+--oracle.xml.async.XSLTransformer
```

## All Implemented Interfaces:

```
java.lang.Runnable, java.io.Serializable,  
oracle.xml.async.XSLTransformerConstants
```

## Description

Applies XSL transformation in a background thread.

## Fields

### methodToCall

```
protected int methodToCall  
The XSL transformation method to call based on input types.
```

### result

```
protected oracle.xml.async.DocumentFragment result  
Transformation result document.
```

## Constructors

### XSLTransformer()

```
public XSLTransformer()  
XSLTransformer constructor
```

### XSLTransformer(int)

```
public XSLTransformer(int id)  
XSLTransformer constructor accepting an identifier
```

**Parameters:**

[id](#) - - A unique integer that can be used to identify the XSLTransformer instance during event processing

**Methods****addXSLTransformerErrorListener(XSLTransformerErrorListener)**

```
public void addXSLTransformerErrorListener(XSLTransformerErrorListener p0)
```

Adds an XSLTransformer error event listener

**Parameters:**

[p0](#) - XSLTransformerErrorListener to be added

**addXSLTransformerListener(XSLTransformerListener)**

```
public void addXSLTransformerListener(XSLTransformerListener p0)
```

Adds a XSLTransformer listener

**Parameters:**

[p0](#) - XSLTransformerListener to be added

**getId()**

```
public int getId()
```

Returns the unique XSLTransformer id

**Returns:**

The id of this XSLTransformer.

**getResult()**

```
public synchronized oracle.xml.async.DocumentFragment getResult()
```

Returns the document fragment for the resulting document. Call this method only after receiving notification that the transformation is complete. Since the transformation occurs in background and asynchronously, calling this method immediately after processXSL will result in holding the control until the result is available.

**Returns:**

The resulting document fragment of the XSL transformation.

### **processXSL(XSLStylesheet, InputStream, URL)**

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.io.InputStream  
xml, java.net.URL ref)
```

Initiates XSL Transformation in the background. The control is returned immediately.

#### **Parameters:**

[xsl](#) - The stylesheet to be used for XSL transformation

[xml](#) - The XML document to be used (as a java.io.InputStream)

[ref](#) - Reference URL to resolve external entities in input XML

#### **Throws:**

[XSLException](#) - if an error occurs during XSL transformation

### **processXSL(XSLStylesheet, Reader, URL)**

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.io.Reader xml,  
java.net.URL ref)
```

Initiates XSL Transformation in the background. The control is returned immediately.

#### **Parameters:**

[xsl](#) - The stylesheet to be used for XSL transformation

[xml](#) - The XML document to be used (as a java.io.Reader)

[ref](#) - Reference URL to resolve external entities in input XML

#### **Throws:**

[XSLException](#) - if an error occurs during XSL transformation

### **processXSL(XSLStylesheet, URL, URL)**

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.net.URL xml,  
java.net.URL ref)
```

Initiates XSL Transformation in the background. The control is returned immediately.

#### **Parameters:**

[xsl](#) - The stylesheet to be used for XSL transformation

[xml](#) - The XML document to be used (as a java.net.URL)

[ref](#) - Reference URL to resolve external entities in input XML

**Throws:**

[XSLException](#) - if an error occurs during XSL transformation

**processXSL(XSLStylesheet, XMLDocument)**

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl,  
oracle.xml.async.XMLDocument xml)
```

Initiates XSL Transformation in the background. The control is returned immediately.

**Parameters:**

[xsl](#) - The stylesheet to be used for XSL transformation

[xml](#) - The XML document to be used (as a DOM Tree)

**Throws:**

[XSLException](#) - if an error occurs during XSL transformation

**processXSL(XSLStylesheet, XMLDocument, OutputStream)**

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl,  
oracle.xml.async.XMLDocument xml, java.io.OutputStream os)
```

Initiates XSL Transformation in the background. The control is returned immediately.

**Parameters:**

[xsl](#) - The stylesheet to be used for XSL transformation

[xml](#) - The XML document to be used (as a DOM Tree)

[os](#) - Outputstream to which the XSL transformation result is written

**Throws:**

[XSLException](#) - if an error occurs during XSL transformation

**removeDOMTransformerErrorListener(XSLTransformerErrorListener)**

```
public synchronized void  
removeDOMTransformerErrorListener(XSLTransformerErrorListener p0)
```



Removes an XSLTransformer error event listener

**Parameters:**

[p0](#) - XSLTransformerErrorListener to be removed

**removeXSLTransformerListener(XSLTransformerListener)**

```
public synchronized void removeXSLTransformerListener(XSLTransformerListener p0)
```

Removes a XSLTransformer listener

**Parameters:**

[p0](#) - XSLTransformerListener to be removed

**run()**

```
public void run()
```

Starts a separate thread to do the XSL Transformation.

**Specified By:**

java.lang.Runnable.run() in interface java.lang.Runnable

**setErrorStream(OutputStream)**

```
public final void setErrorStream(java.io.OutputStream out)
```

Sets the error stream used by the XSL processor

**Parameters:**

[out](#) - The error output stream for the XSL processor

**showWarnings(boolean)**

```
public final void showWarnings(boolean yes)
```

Sets the showWarnings flag used by the XSL processor

**Parameters:**

[yes](#) - Boolean indicating if XSL processor warnings to be shown or not.

## XSLTransformerBeanInfo

### Syntax

```
public class XSLTransformerBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
|
+--java.beans.SimpleBeanInfo
|
+--oracle.xml.async.XSLTransformerBeanInfo
```

### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Description

This class provides information about the XSLTransformer Bean.

## Constructors

### XSLTransformerBeanInfo()

```
public XSLTransformerBeanInfo()
```

The default Constructor

## Methods

### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

Gets an image object that can be used to represent [XSLTransformer](#) bean in toolbars, toolboxes, etc.

### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

### Parameters:

[iconKind](#) - The kind of icon requested.

**Returns:**

An image object representing the requested icon type for [XSLTransformer](#) bean.

**getPropertyDescriptors()**

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

Gets the [XSLTransformer](#) bean's [PropertyDescriptors](#)

**Overrides:**

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class  
java.beans.SimpleBeanInfo

**Returns:**

An array of PropertyDescriptors describing the editable properties supported by [XSLTransformer](#) bean.

## XSLTransformerErrorEvent

### Syntax

```
public class XSLTransformerErrorEvent extends java.util.EventObject
```

```
java.lang.Object  
|  
+--java.util.EventObject  
|  
+--oracle.xml.async.XSLTransformerErrorEvent
```

### All Implemented Interfaces:

```
java.io.Serializable
```

### Description

The error event object that XSLTransformer uses to notify all registered listeners about transformation error events.

### Fields

```
protected java.lang.Exception e  
The exception being raised.
```

### Constructors

#### XSLTransformerErrorEvent(Object, Exception)

```
public XSLTransformerErrorEvent(java.lang.Object p0, java.lang.Exception e)  
Constructor for XSLTransformerErrorEvent.
```

#### Parameters:

[p0](#) - The [Object](#) that created this event

[e](#) - The [Exception](#) raised.

### Methods

#### getException()

```
public java.lang.Exception getException()
```

Returns the exception that XSLTransformer encountered object unique id. Can be used to

**Returns:**

The transformation exception

**getMessage()**

```
public java.lang.String getMessage()
```

Returns the error message that describes the error that XSLTransformer encountered

**Returns:**

The error message

## XSLTransformerErrorListener

### Syntax

```
public interface XSLTransformerErrorListener extends java.util.EventListener
```

### All Superinterfaces:

```
java.util.EventListener
```

### Description

This interface must be implemented in order to receive notifications about error events during the asynchronous transformation. The class implementing this interface must be added to the XSLTransformer using `addXSLTransformerListener` method.

### Methods

#### `xslTransformerErrorCalled(XSLTransformerErrorEvent)`

```
public void xslTransformerErrorCalled(XSLTransformerErrorEvent p0)
```

This method is called when parse or transformation error occurs.

#### Parameters:

[p0](#) - - The XSLTransformerErrorEvent object produced by the XSLTransformer

## XSLTransformerEvent

### Syntax

```
public class XSLTransformerEvent extends java.util.EventObject
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--oracle.xml.async.XSLTransformerEvent
```

### All Implemented Interfaces:

```
java.io.Serializable
```

### Fields

#### id

```
protected int id
ID of the source XSLTransformer object
```

### Constructors

#### XSLTransformerEvent(Object, int)

```
public XSLTransformerEvent(java.lang.Object p0, int p1)
```

Constructs the XSLTransformerEvent object using the XSLTransformer source object and its unique id.

#### Parameters:

[`<code>p0</code>`](#) - The source XSLTransformer object that will fire the events

[`<code>p1</code>`](#) - Unique id identifying the source object

### Methods

#### getID()

```
public int getID()
```

Returns unique id of the XSLTransformer object which can be used to identify which instance of the XSLTransformer generated this event in cases where multiple instances of XSLTransformer may be working in background.

**Returns:**

The unique [id](#) of the source XSLTransformer object for this event object.



## XSLTransformerListener

### Syntax

```
public interface XSLTransformerListener extends java.util.EventListener
```

### All Superinterfaces:

```
java.util.EventListener
```

### Description

This interface must be implemented in order to receive notifications about events during the asynchronous transformation. The class implementing this interface must be added to the XSLTransformer using addXSLTransformerListener method.

### Methods

#### xslTransformerError(XSLTransformerEvent)

```
public void xslTransformerError(XSLTransformerEvent p0)
```

This method is called when parse or transformation error occur.

#### Parameters:

[p0](#) - - The XSLTransformerEvent object produced by the XSLTransformer

#### xslTransformerOver(XSLTransformerEvent)

```
public void xslTransformerOver(XSLTransformerEvent p0)
```

This method is called when the transformation is complete

#### Parameters:

[p0](#) - - The XSLTransformerEvent object produced by the XSLTransformer

#### xslTransformerStarted(XSLTransformerEvent)

```
public void xslTransformerStarted(XSLTransformerEvent p0)
```

This method is called when the transformation starts

#### Parameters:

[p0](#) - - The XSLTransformerEvent object produced by the XSLTransformer

## Package oracle.xml.dbviewer

### DBViewer

#### Syntax

```
public class DBViewer extends javax.swing.JPanel implements java.io.Serializable
```

```
java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|
+--javax.swing.JPanel
|
+--oracle.xml.dbviewer.DBViewer
```

#### All Implemented Interfaces:

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

#### Description

Java bean that can be used to display database queries or any XML by applying XSL stylesheets and visualizing the resulted HTML in scrollable swing panel. This bean has tree buffers: XML, XSL and result buffer. The bean API allow the calling program to load/save the buffers from various sources and to apply stylesheet transformation to the XML buffer using the stylesheet in the XSL buffer. The result can be stored in the result buffer. The XML and XSL buffers content can be shown as source or as a tree structure. The result buffer content can be rendered as HTML and also shown as source or tree structure. The XML buffer can be loaded from database query. All buffers can load and save files from CLOB tables in Oracle database and from file system as well. Therefore, the control can be also used to move files between the file system and the user schema in the database.

## Constructors

### DBViewer()

```
public DBViewer()  
Constructs a new instance.
```

## Methods

### getHostname()

```
public java.lang.String getHostname()  
Get database host name
```

**Returns:**  
host name

### getInstancename()

```
public java.lang.String getInstancename()  
Get database instance name
```

**Returns:**  
database instance name

### getPassword()

```
public java.lang.String getPassword()  
Get user password
```

**Returns:**  
user password

### getPort()

```
public java.lang.String getPort()  
Get database port number
```

**Returns:**  
String with the database port number

### **getResBuffer()**

```
public java.lang.String getResBuffer()  
Get the content of the result buffer
```

**Returns:**  
the buffer content

### **getResCLOBFileName()**

```
public java.lang.String getResCLOBFileName()  
Get result CLOB file name
```

**Returns:**  
result CLOB file name

### **getResCLOBTableName()**

```
public java.lang.String getResCLOBTableName()  
Get result CLOB table name
```

**Returns:**  
result CLOB table name

### **getResFileName()**

```
public java.lang.String getResFileName()  
Get Result file name
```

**Returns:**  
XSL file name

### **getUsername()**

```
public java.lang.String getUsername()  
Get user name
```

**Returns:**  
user name

**getXmlBuffer()**

```
public java.lang.String getXmlBuffer()  
Get the content of the XML buffer
```

**Returns:**

the buffer content

**getXmlCLOBFileName()**

```
public java.lang.String getXmlCLOBFileName()  
Get XML CLOB file name
```

**Returns:**

XML CLOB file name

**getXmlCLOBTableName()**

```
public java.lang.String getXmlCLOBTableName()  
Get XML CLOB table name
```

**Returns:**

XML CLOB table name

**getXmlFileName()**

```
public java.lang.String getXmlFileName()  
Get XML file name
```

**Returns:**

XML file name

**getXMLStringFromSQL(String)**

```
public java.lang.String getXMLStringFromSQL(java.lang.String sqlText)  
Get XML presentation of result set from SQL query
```

**Returns:**

the query result set as XML string

**getXslBuffer()**

```
public java.lang.String getXslBuffer()  
Get the content of the XSL buffer
```

**Returns:**

the buffer content

**getXslCLOBFileName()**

```
public java.lang.String getXslCLOBFileName()  
Get the XSL CLOB file name
```

**Returns:**

XSL CLOB file name

**getXslCLOBTableName()**

```
public java.lang.String getXslCLOBTableName()  
Get XSL CLOB table name
```

**Returns:**

XSL CLOB table name

**getXslFileName()**

```
public java.lang.String getXslFileName()  
Get XSL file name
```

**Returns:**

XSL file name

**loadResBuffer(String)**

```
public void loadResBuffer(java.lang.String filename)  
Load the result buffer from file
```

**Parameters:**

[filename](#) - file name

### loadResBuffer(String, String)

```
public void loadResBuffer(java.lang.String tablename, java.lang.String filename)
```

Load the result buffer from CLOB file

**Parameters:**

[tablename](#) - CLOB table name

[filename](#) - CLOB file name

### loadResBuffer(XMLDocument)

```
public void loadResBuffer(oracle.xml.parser.v2.XMLDocument resdoc)
```

Load the result buffer from XMLDocument

**Parameters:**

[resdoc](#) -- the XMLDocument

### loadResBufferFromClob()

```
public void loadResBufferFromClob()
```

Load the result buffer from CLOB file

### loadResBufferFromFile()

```
public void loadResBufferFromFile()
```

Load the result buffer from file

### loadXmlBuffer(String)

```
public void loadXmlBuffer(java.lang.String filename)
```

Load the XML buffer from file

**Parameters:**

[filename](#) - file name

### loadXmlBuffer(String, String)

```
public void loadXmlBuffer(java.lang.String tablename, java.lang.String filename)
```

Load the XML buffer from CLOB file

**Parameters:**

[tablename](#) - CLOB table name

[filename](#) - CLOB file name

### loadXmlBuffer(XMLDocument)

```
public void loadXmlBuffer(oracle.xml.parser.v2.XMLDocument xmldoc)
```

Load the XML buffer from XMLDocument

#### Parameters:

[filename](#) - file name

### loadXmlBufferFromClob()

```
public void loadXmlBufferFromClob()
```

Load the XML buffer from CLOB file

### loadXmlBufferFromFile()

```
public void loadXmlBufferFromFile()
```

Load the XML buffer from file

### loadXMLBufferFromSQL(String)

```
public void loadXMLBufferFromSQL(java.lang.String sqltext)
```

Load the XML buffer from SQL result set

#### Parameters:

[sqltext](#) - SQL text

### loadXslBuffer(String)

```
public void loadXslBuffer(java.lang.String filename)
```

Load the XSL buffer from file

#### Parameters:

[filename](#) - file name

### loadXslBuffer(String, String)

```
public void loadXslBuffer(java.lang.String tablename, java.lang.String filename)
```

Load the XSL buffer from CLOB file

#### Parameters:

[tablename](#) - CLOB table name



---

[filename](#) - CLOB file name

### loadXslBuffer(XMLDocument)

```
public void loadXslBuffer(oracle.xml.parser.v2.XMLDocument xslDoc)
```

Load the XSL buffer from XMLDocument

#### Parameters:

[xslDoc](#) -- the XML Document

### loadXslBufferFromClob()

```
public void loadXslBufferFromClob()
```

Load the XSL buffer from CLOB file

### loadXslBufferFromFile()

```
public void loadXslBufferFromFile()
```

Load the XSL buffer from file

### parseResBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseResBuffer()
```

Parse the result buffer and refresh the tree view and source view

#### Returns:

XMLDocument

### parseXmlBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseXmlBuffer()
```

Parse the XML buffer and refresh the tree view and source view

#### Returns:

XMLDocument

### parseXslBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseXslBuffer()
```

Parse the XSL buffer and refresh the tree view and source view

**Returns:**

XMLDocument

**saveResBuffer(String)**

```
public void saveResBuffer(java.lang.String filename)
```

Save the result buffer to file

**Parameters:**[filename](#) - CLOB file name**saveResBuffer(String, String)**

```
public void saveResBuffer(java.lang.String tablename, java.lang.String filename)
```

Save the result buffer to CLOB file

**Parameters:**[tablename](#) - CLOB table name[filename](#) - CLOB file name**saveResBufferToClob()**

```
public void saveResBufferToClob()
```

Save the result buffer to CLOB file

**saveResBufferToFile()**

```
public void saveResBufferToFile()
```

Save the result buffer to file

**saveXmlBuffer(String)**

```
public void saveXmlBuffer(java.lang.String filename)
```

Save the XML buffer to file

**Parameters:**[filename](#) - file name**saveXmlBuffer(String, String)**

```
public void saveXmlBuffer(java.lang.String tablename, java.lang.String filename)
```

Save the XML buffer to CLOB file

**Parameters:**

[tablename](#) - CLOB table name

[filename](#) - CLOB file name

**saveXmlBufferToClob()**

```
public void saveXmlBufferToClob()
```

Save the XML buffer to CLOB file

**saveXmlBufferToFile()**

```
public void saveXmlBufferToFile()
```

Save the XML buffer to file

**saveXslBuffer(String)**

```
public void saveXslBuffer(java.lang.String filename)
```

Save the XSL buffer to file

**Parameters:**

[filename](#) - file name

**saveXslBuffer(String, String)**

```
public void saveXslBuffer(java.lang.String tablename, java.lang.String filename)
```

Save the XSL buffer to CLOB file

**Parameters:**

[tablename](#) - CLOB table name

[filename](#) - CLOB file name

**saveXslBufferToClob()**

```
public void saveXslBufferToClob()
```

Save the XSL buffer to CLOB file

**saveXslBufferToFile()**

```
public void saveXslBufferToFile()
```

Save the XSL buffer to file

**setHostname(String)**

```
public void setHostname(java.lang.String hostname)
Set database host name
```

**Parameters:**

[hostname](#) - the host name

**setInstancename(String)**

```
public void setInstancename(java.lang.String instancename)
Set database instance name
```

**Parameters:**

[instancename](#) - the database instance name

**setPassword(String)**

```
public void setPassword(java.lang.String password)
Set user password
```

**Parameters:**

[password](#) - the user password

**setPort(String)**

```
public void setPort(java.lang.String port)
Set database port number
```

**Parameters:**

[port](#) - String containing the port number

**setResBuffer(String)**

```
public void setResBuffer(java.lang.String text)
Set new text in the result buffer
```

**Parameters:**

[text](#) - the new text

**setResCLOBFileName(String)**

```
public void setResCLOBFileName(java.lang.String name)
Set Result CLOB file name
```

**Parameters:**

[name](#) - Result CLOB file name

**setResCLOBTableName(String)**

```
public void setResCLOBTableName(java.lang.String name)
Set Result CLOB table name
```

**Parameters:**

[name](#) - Result CLOB table name

**setResFileName(String)**

```
public void setResFileName(java.lang.String name)
Set Result file name
```

**Parameters:**

[name](#) - Result file name

**setResHtmlView(boolean)**

```
public void setResHtmlView(boolean on)
Show the result buffer as rendered HTML
```

**setResSourceEditView(boolean)**

```
public void setResSourceEditView(boolean on)
Show the result buffer as XML source and enter edit mode
```

**setResSourceView(boolean)**

```
public void setResSourceView(boolean on)
Show the result buffer as XML source
```

**setResTreeView(boolean)**

```
public void setResTreeView(boolean on)
Show the result buffer as XML tree view
```

**setUsername(String)**

```
public void setUsername(java.lang.String username)
Set user name
```

**Parameters:**

[username](#) - the user name

**setXmlBuffer(String)**

```
public void setXmlBuffer(java.lang.String text)
Set new text in the XML buffer
```

**Parameters:**

[text](#) - XML text

**setXmlCLOBFileName(String)**

```
public void setXmlCLOBFileName(java.lang.String name)
Set XML CLOB table name
```

**Parameters:**

[name](#) - XML CLOB table name

**setXmlCLOBTableName(String)**

```
public void setXmlCLOBTableName(java.lang.String name)
Set XML CLOB table name
```

**Parameters:**

[name](#) - XML CLOB table name

**setXmlFileName(String)**

```
public void setXmlFileName(java.lang.String name)
Set XML file name
```

**Parameters:**

[name](#) - XML file name

**setXmlSourceEditView(boolean)**

```
public void setXmlSourceEditView(boolean on)
Show the XML buffer as XML source and enter edit mode
```

**setXmlSourceView(boolean)**

```
public void setXmlSourceView(boolean on)
Show the XML buffer as XML source
```

**setXmlTreeView(boolean)**

```
public void setXmlTreeView(boolean on)
Show the XML buffer as tree
```

**setXslBuffer(String)**

```
public void setXslBuffer(java.lang.String text)
Set new text in the XSL buffer
```

**Parameters:**

[text](#) - XSL text

**setXslCLOBFileName(String)**

```
public void setXslCLOBFileName(java.lang.String name)
Set XSL CLOB file name
```

**Parameters:**

[name](#) - XSL CLOB file name

**setXslCLOBTableName(String)**

```
public void setXslCLOBTableName(java.lang.String name)
Set XSL CLOB table name
```

**Parameters:**

[name](#) - XSL CLOB table name

**setXslFileName(String)**

```
public void setXslFileName(java.lang.String name)
Set XSL file name
```

**Parameters:**

[name](#) - XSL file name

**setXslSourceEditView(boolean)**

```
public void setXslSourceEditView(boolean on)
```

Show the XSL buffer as XML source and enter edit mode

**setXslSourceView(boolean)**

```
public void setXslSourceView(boolean on)
```

Show the XSL buffer as XML source

**setXslTreeView(boolean)**

```
public void setXslTreeView(boolean on)
```

Show the XSL buffer as tree

**transformToDoc()**

```
public oracle.xml.parser.v2.XMLDocument transformToDoc()
```

Transforms the content of the XML buffer by applying the stylesheet from the XSL buffer.

**transformToRes()**

```
public void transformToRes()
```

Apply the stylesheet transformation from the XSL buffer to the XML in the XML buffer and stores the result into the result buffer

**transformToString()**

```
public java.lang.String transformToString()
```

Transforms the content of the XML buffer by applying the stylesheet from the XSL buffer.



## Package oracle.xml.dbviewer

### DBViewerBeanInfo

#### Syntax

```
public class DBViewerBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
```

```
|
```

```
+--java.beans.SimpleBeanInfo
```

```
|
```

```
+--oracle.xml.dbviewer.DBViewerBeanInfo
```

#### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Constructors

#### DBViewerBeanInfo()

```
public DBViewerBeanInfo()
```

Constructor

### Methods

#### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

#### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

#### getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

#### Overrides:

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class  
java.beans.SimpleBeanInfo

## Package oracle.xml.srcviewer

### XMLSourceView

#### Syntax

```
public class XMLSourceView extends javax.swing.JPanel implements
java.io.Serializable
```

```
java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|
+--javax.swing.JPanel
|
+--oracle.xml.srcviewer.XMLSourceView
```

#### All Implemented Interfaces:

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

#### Description

Shows an XML document. Recognizes the following XML token types: [Tag](#), [Attribute Name](#), [Attribute Value](#), [Comment](#), [CDATA](#), [PCDATA](#), [PI Data](#), [PI Name and NOTATION Symbol](#). Each token type has a foreground color and font. The default color/font settings can be changed by the user. Takes as input an org.w3c.dom.Document object.

#### Fields

##### inputDOMDocument

```
protected org.w3c.dom.Document inputDOMDocument
```

##### jScrollPane

```
protected javax.swing.JScrollPane jScrollPane
```

## JTextPane

```
protected javax.swing.JTextPane JTextPane
```

## xmlStyledDocument

```
protected oracle.xml.srcviewer.XMLStyledDocument xmlStyledDocument
```

## Constructors

### XMLSourceView()

```
public XMLSourceView()
```

The class constructor. Creates an object of type [XMLSourceView](#).

## Methods

### fontGet(AttributeSet)

```
public static java.awt.Font fontGet(javax.swing.text.AttributeSet attributeset)
```

Extracts and returns the font from a given attributeset.

#### Parameters:

[attributeset](#) - The source [Attributeset](#).

#### Returns:

The extracted [Font](#).

### fontSet(MutableAttributeSet, Font)

```
public static void fontSet(javax.swing.text.MutableAttributeSet mutableattributeset, java.awt.Font font)
```

Sets the mutableattributeset font.

#### Parameters:

[mutableattributeset](#) - The [mutableattributeset](#) to update.

[font](#) - The new [Font](#) for the mutableattributeset.

### getAttributeNameFont()

```
public java.awt.Font getAttributeNameFont()
```

Returns the Attribute Value font.

**Returns:**

The [Font](#) object.

**getAttributeNameForeground()**

```
public java.awt.Color getAttributeNameForeground()  
Returns the Attribute Name foreground color.
```

**Returns:**

The [Color](#) object.

**getAttributeValueFont()**

```
public java.awt.Font getAttributeValueFont()  
Returns the Attribute Value font.
```

**Returns:**

The [Font](#) object.

**getAttributeValueForeground()**

```
public java.awt.Color getAttributeValueForeground()  
Returns the Attribute Value foreground color.
```

**Returns:**

The [Color](#) object.

**getBackground()**

```
public java.awt.Color getBackground()  
Returns the background color.
```

**Overrides:**

java.awt.Component.getBackground() in class java.awt.Component

**Returns:**

The [Color](#) object.

**getCDATAFont()**

```
public java.awt.Font getCDATAFont()
```

Returns the CDATA font.

**Returns:**

The [Font](#) object.

**getCDATAForeground()**

```
public java.awt.Color getCDATAForeground()
```

Returns the CDATA foreground color.

**Returns:**

The [Color](#) object.

**getCommentDataFont()**

```
public java.awt.Font getCommentDataFont()
```

Returns the Comment Data font.

**Returns:**

The [Font](#) object.

**getCommentDataForeground()**

```
public java.awt.Color getCommentDataForeground()
```

Returns the Comment Data foreground color.

**Returns:**

The [Color](#) object.

**getEditedText()**

```
public java.lang.String getEditedText()
```

Returns the edited text.

**Returns:**

The [String](#) object containing the edited text.

**getJTextPane()**

```
public javax.swing.JTextPane getJTextPane()
```

Returns the viewer [JTextPane](#) component.

**Returns:**

The [JTextPane](#) object used by XMLSourceViewer

**getMinimumSize()**

```
public java.awt.Dimension getMinimumSize()
```

Returns the XMLSourceView minimal size.

**Overrides:**

javax.swing.JComponent.getMinimumSize() in class javax.swing.JComponent

**Returns:**

The [Dimension](#) object containing the XMLSourceView minimum size.

**getNodeAtOffset(int)**

```
public org.w3c.dom.Node getNodeAtOffset(int i)
```

Returns the XML node at a given offset.

**Parameters:**

[i](#) - The node offset.

**Returns:**

The [Node](#) object from offset [i](#).

**getPCDATAFont()**

```
public java.awt.Font getPCDATAFont()
```

Returns the PCDATA font.

**Returns:**

The [Font](#) object.

**getPCDATAForeground()**

```
public java.awt.Color getPCDATAForeground()
```

Returns the PCDATA foreground color.

**Returns:**

The [Color](#) object.

**getPIDataFont()**

```
public java.awt.Font getPIDataFont()
```

Returns the PI Data font.

**Returns:**

The [Font](#) object

**getPIDataForeground()**

```
public java.awt.Color getPIDataForeground()
```

Returns the PI Data foreground color.

**Returns:**

The [Color](#) object.

**getPINameFont()**

```
public java.awt.Font getPINameFont()
```

Returns the PI Name font.

**Returns:**

The [Font](#) object.

**getPINameForeground()**

```
public java.awt.Color getPINameForeground()
```

Returns the PI Data foreground color.

**Returns:**

The [Color](#) object.

**getSymbolFont()**

```
public java.awt.Font getSymbolFont()
```

Returns the NOTATION Symbol font.

**Returns:**

The [Font](#) object.

**getSymbolForeground()**

```
public java.awt.Color getSymbolForeground()
Returns the NOTATION Symbol foreground color.
```

**Returns:**

The [Color](#) object.

**getTagFont()**

```
public java.awt.Font getTagFont()
Returns the Tag font.
```

**Returns:**

The [Font](#) object.

**getTagForeground()**

```
public java.awt.Color getTagForeground()
Returns the Tag foreground color.
```

**Returns:**

The [Color](#) object.

**getText()**

```
public java.lang.String getText()
Returns the XML document as a String.
```

**Returns:**

The [String](#) object containing the XML document.

**isEditable()**

```
public boolean isEditable()
Returns boolean to indicate whether this object is editable.
```

**selectNodeAt(int)**

```
public void selectNodeAt(int i)
Moves the cursor to XML Node at offset i.
```



**Parameters:**

[i](#) - The node offset.

**setAttributeNameFont(Font)**

```
public void setAttributeNameFont(java.awt.Font font)
```

Sets the Attribute Name font.

**Parameters:**

[font](#) - The new [Font](#) for Attribute Name.

**setAttributeNameForeground(Color)**

```
public void setAttributeNameForeground(java.awt.Color color)
```

Sets the Attribute Name foreground color.

**Parameters:**

[color](#) - The new [Color](#) for Attribute Name.

**setAttributeValueFont(Font)**

```
public void setValueFont(java.awt.Font font)
```

Sets the Attribute Value font.

**Parameters:**

[font](#) - The new [Font](#) for Attribute Value.

**setAttributeValueForeground(Color)**

```
public void setValueForeground(java.awt.Color color)
```

Sets the Attribute Value foreground color.

**Parameters:**

[color](#) - The new [Color](#) for Attribute Value.

**setBackground(Color)**

```
public void setBackground(java.awt.Color color)
```

Sets the background color.

**Overrides:**

javax.swing.JComponent.setBackground(java.awt.Color) in class  
javax.swing.JComponent

**Parameters:**

[color](#) - The new background [Color](#).

**setCDATAFont(Font)**

```
public void setCDATAFont(java.awt.Font font)
```

Sets the CDATA font.

**Parameters:**

[font](#) - The new [Font](#) for CDATA.

**setCDATAForeground(Color)**

```
public void setCDATAForeground(java.awt.Color color)
```

Sets the CDATA foreground color.

**Parameters:**

[color](#) - The new [Color](#) for CDATA.

**setCommentDataFont(Font)**

```
public void setCommentDataFont(java.awt.Font font)
```

Sets the Comment font.

**Parameters:**

[font](#) - The new [Font](#) for the XML Comments.

**setCommentDataForeground(Color)**

```
public void setCommentDataForeground(java.awt.Color color)
```

Sets the Comment foreground color.

**Parameters:**

[color](#) - The new [Color](#) for Comment.

**setEditable(boolean)**

```
public void setEditable(boolean edit)
```

Sets the specified boolean to indicate whether this object should be editable.

**Parameters:**

[doc](#) - The new [boolean](#) value.

**setPCDATAFont(Font)**

```
public void setPCDATAFont(java.awt.Font font)
```

Sets the PCDATA font.

**Parameters:**

[font](#) - The new [Font](#) for PCDATA.

**setPCDATAForeground(Color)**

```
public void setPCDATAForeground(java.awt.Color color)
```

Sets the PCDATA foreground color.

**Parameters:**

[color](#) - The new [Color](#) for PCDATA.

**setPIDataFont(Font)**

```
public void setPIDataFont(java.awt.Font font)
```

Sets the PI Data font.

**Parameters:**

[font](#) - The new [Font](#) for PI Data.

**setPIDataForeground(Color)**

```
public void setPIDataForeground(java.awt.Color color)
```

Sets the PI Data foreground color.

**Parameters:**

[color](#) - The new [Color](#) for PI Data.

### setPNameFont(Font)

```
public void setPNameFont(java.awt.Font font)
```

Sets the PI Name font.

**Parameters:**

[font](#) - The new [Font](#) for the PI Names.

### setPNameForeground(Color)

```
public void setPNameForeground(java.awt.Color color)
```

Sets the PI Name foreground color.

**Parameters:**

[color](#) - The new [Color](#) for PI Name.

### setSelectedNode(Node)

```
public void setSelectedNode(org.w3c.dom.Node node)
```

Sets the cursor position at the selected XML node.

**Parameters:**

[node](#) - The selected node.

### setSymbolFont(Font)

```
public void setSymbolFont(java.awt.Font font)
```

Sets the NOTATION Symbol font.

**Parameters:**

[color](#) - The new [Font](#) for NOTATION Symbol.

### setSymbolForeground(Color)

```
public void setSymbolForeground(java.awt.Color color)
```

Sets the NOTATION Symbol foreground color.

**Parameters:**

[color](#) - The new [Color](#) for NOTATION Symbol.

### setTagFont(Font)

```
public void setTagFont(java.awt.Font font)
```

Sets the Tag font.

**Parameters:**

[font](#) - The new [Font](#) for the XML Tags.

### setTagForeground(Color)

```
public void setTagForeground(java.awt.Color color)
```

Sets the Tag foreground color.

**Parameters:**

[color](#) - The new [Color](#) for the XML Tags.

### setXMLDocument(Document)

```
public void setXMLDocument(org.w3c.dom.Document document)
```

Associates the XMLviewer with a XML document.

**Parameters:**

[doc](#) - The [Document](#) document to display.

**See Also:**

[getText\(\)](#)

## Package oracle.xml.srcviewer

### XMLSourceViewBeanInfo

#### Syntax

```
public class XMLSourceViewBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
|
+--java.beans.SimpleBeanInfo
|
+--oracle.xml.srcviewer.XMLSourceViewBeanInfo
```

#### All Implemented Interfaces:

```
java.beans.BeanInfo
```

#### Constructors

##### XMLSourceViewBeanInfo()

```
public XMLSourceViewBeanInfo()
```

#### Methods

##### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

#### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

##### getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

#### Overrides:

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class java.beans.SimpleBeanInfo

## Package oracle.xml.transviewer

### DBAccess

#### Syntax

```
public class DBAccess extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.xml.transviewer.DBAccess
```

#### Description

Maintains CLOB tables that can hold multiple XML and text documents. Each table is created using the statement: `CREATE TABLE tablename FILENAME CHAR( 16) UNIQUE, FILEDATA CLOB) LOB(FILEDATA) STORE AS (DISABLE STORAGE IN ROW)`. Each XML (or text) document is stored as a row in the table and the FILENAME field holds a unique string that is used as a key to retrieve, update or delete the row. The document text is stored in the FILEDATA field that is a CLOB object. This CLOB tables are automatically maintained by the transviewer bean. The CLOB tables maintained by this class can be later used by the transviewer bean. The class creates and deletes CLOB tables, list a CLOB table content and also add, replace or delete text documents in this CLOB tables.

#### Constructors

##### DBAccess()

```
public DBAccess()
```

#### Methods

##### createBLOBTable(Connection, String)

```
public boolean createBLOBTable(java.sql.Connection con, java.lang.String  
tablename)
```

Create BLOB table

##### Parameters:

[con](#) -- the Connection object

[tablename](#) -- the table name

**Returns:**

true if successful

**createXMLTable(Connection, String)**

```
public boolean createXMLTable(java.sql.Connection con, java.lang.String
tablename)
```

Create XML table

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

**Returns:**

true if successful

**deleteBLOBName(Connection, String, String)**

```
public boolean deleteBLOBName(java.sql.Connection con, java.lang.String
tablename, java.lang.String xmlname)
```

Delete binary file from BLOB table

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

[xmlname](#) -- the file name

**Returns:**

true if successful

**deleteXMLName(Connection, String, String)**

```
public boolean deleteXMLName(java.sql.Connection con, java.lang.String
tablename, java.lang.String xmlname)
```

Delete file from XML table



**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

[xmlname](#) -- the file name

**Returns:**

true if successful

**dropBLOBTable(Connection, String)**

```
public boolean dropBLOBTable(java.sql.Connection con, java.lang.String  
tablename)
```

Delete BLOB table

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

**Returns:**

true if successful

**dropXMLTable(Connection, String)**

```
public boolean dropXMLTable(java.sql.Connection con, java.lang.String tablename)
```

Delete XML table

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

**Returns:**

true if successful

**getBLOBData(Connection, String, String)**

```
public byte[] getBLOBData(java.sql.Connection con, java.lang.String tablename,  
java.lang.String xmlname)
```

Retrieve binary file from BLOB table

**Parameters:**

[con](#) - - the Connection object

[tablename](#) - - the table name

[xmlname](#) - - the file name

**Returns:**

file as a byte array

**getNameSize()**

```
public int getNameSize()
```

Returns the size of the field where the filename is kept.

**Returns:**

filename size

**getXMLData(Connection, String, String)**

```
public java.lang.String getXMLData(java.sql.Connection con, java.lang.String  
tablename, java.lang.String xmlname)
```

Retrieve text file from XML table

**Parameters:**

[con](#) - - the Connection object

[tablename](#) - - the table name

[xmlname](#) - - the file name

**Returns:**

file as a string

**getXMLNames(Connection, String)**

```
public java.lang.String[] getXMLNames(java.sql.Connection con, java.lang.String  
tablename)
```

Returns all file names in XML table

**Parameters:**

[con](#) - - the Connection object

[tablename](#) - - the table name

**Returns:**

String array with all file names in this table

**getXMLTableNames(Connection, String)**

```
public java.lang.String[] getXMLTableNames(java.sql.Connection con,  
java.lang.String tablePrefix)
```

Gets all XML tables with names starting with a given string

**Parameters:**

[con](#) - - the Connection object

[tablePrefix](#) - - table prefix string

**Returns:**

array of all XML tables that begin with tablePrefix

**insertBLOBData(Connection, String, String, byte[])**

```
public boolean insertBLOBData(java.sql.Connection con, java.lang.String  
tablename, java.lang.String xmlname, byte[] xmldata)
```

Inserts binary file as a row in BLOB table

**Parameters:**

[con](#) - - the Connection object

[tablename](#) - - the table name

[xmlname](#) - - the file name

[xmldata](#) - - byte array with file data

**Returns:**

true if successful

**insertXMLData(Connection, String, String, String)**

```
public boolean insertXMLData(java.sql.Connection con, java.lang.String  
tablename, java.lang.String xmlname, java.lang.String xmldata)
```

Inserts text file as a row in XML table

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

[xmlname](#) -- the file name

[xmldata](#) -- string with the file data

**Returns:**

true if successfull

**isXMLTable(Connection, String)**

```
public boolean isXMLTable(java.sql.Connection con, java.lang.String tablename)
Check if the table is XML table.
```

**Parameters:**

[con](#) -- the Connection object

[tableName](#) -- the table name to test

**Returns:**

true if this is XML table

**replaceXMLData(Connection, String, String, String)**

```
public boolean replaceXMLData(java.sql.Connection con, java.lang.String
tablename, java.lang.String xmlname, java.lang.String xmldata)
Replace text file as a row in XML table
```

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

[xmlname](#) -- the file name

[xmldata](#) -- string with the file data

**Returns:**

true if successfull

**xmlTableExists(Connection, String)**

```
public boolean xmlTableExists(java.sql.Connection con, java.lang.String  
tablename)
```

Checks if the XML table exists

**Parameters:**

[con](#) -- the Connection object

[tablename](#) -- the table name

**Returns:**

true if the table exists

## DBAccessBeanInfo

### Syntax

```
public class DBAccessBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
```

```
|
```

```
+--java.beans.SimpleBeanInfo
```

```
|
```

```
+--oracle.xml.transviewer.DBAccessBeanInfo
```

### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Constructors

#### DBAccessBeanInfo()

```
public DBAccessBeanInfo()
```

Constructor

### Methods

#### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

#### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

#### getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

#### Overrides:

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class  
java.beans.SimpleBeanInfo

# XMLTransformPanel

## Syntax

```
public class XMLTransformPanel extends javax.swing.JPanel
```

```
java.lang.Object
|
+--java.awt.Component
    |
    +--java.awt.Container
        |
        +--javax.swing.JComponent
            |
            +--javax.swing.JPanel
                |
                +--oracle.xml.transviewer.XMLTransformPanel
```

## All Implemented Interfaces:

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

## Description

XMLTransformPanel visual bean. Applies XSL transformations on XML documents. Visualizes the result. Allows editing of input XML and XSL documents/files.

## Constructors

### XMLTransformPanel()

```
public XMLTransformPanel()
```

The class constructor. Creates an object of type [XMLTransformPanel](#).

## XMLTransformPanelBeanInfo

### Syntax

```
public class XMLTransformPanelBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object  
|  
+--java.beans.SimpleBeanInfo  
|  
+--oracle.xml.transviewer.XMLTransformPanelBeanInfo
```

### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Constructors

#### XMLTransformPanelBeanInfo()

```
public XMLTransformPanelBeanInfo()
```

### Methods

#### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

#### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

#### getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

#### Overrides:

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class java.beans.SimpleBeanInfo



## XMLTransViewer

### Syntax

```
public class XMLTransViewer extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.xml.transviewer.XMLTransViewer
```

### Description

Simple application that uses XMLTransformPanel. Can be used from the command line to edit and parse XML files, edit and apply XSL transformations and retrieve and save XML, XSL and result files in the file system or in the Oracle 8i database

### Constructors

#### XMLTransViewer()

```
public XMLTransViewer()
```

### Methods

#### getReleaseVersion()

```
public static java.lang.String getReleaseVersion()
```

Returns the release version of the Oracle XML Transviewer

#### Returns:

the release version string

#### main(String[])

```
public static void main(java.lang.String[] args)
```

## Package oracle.xml.treeviewer

### XMLTreeView

#### Syntax

```
public class XMLTreeView extends javax.swing.JPanel
```

```
java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|
+--javax.swing.JPanel
|
+--oracle.xml.treeviewer.XMLTreeView
```

#### All Implemented Interfaces:

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,  
java.awt.MenuContainer, java.io.Serializable
```

#### Description

Shows an XML document as a tree. Recognizes the following XML DOM nodes:

[Tag](#), [Attribute Name](#), [Attribute Value](#), [Comment](#), [CDATA](#), [PCDATA](#), [PI Data](#), [PI Name and NOTATION Symbol](#). Takes as input an [org.w3c.dom.Document](#) object.

#### Fields

##### model

```
protected oracle.xml.treeviewer.XMLTreeModel model
```

##### scrollPane

```
protected transient javax.swing.JScrollPane scrollPane
```

**theTree**

```
protected transient javax.swing.JTree theTree
```

**Constructors****XMLTreeView()**

```
public XMLTreeView()
```

The class constructor. Creates an object of type [XMLTreeView](#).

**Methods****getPreferredSize()**

```
public java.awt.Dimension getPreferredSize()
```

Returns the XMLTreeView preferred size.

**Overrides:**

`javax.swing.JComponent.getPreferredSize()` in class `javax.swing.JComponent`

**Returns:**

The [Dimension](#) object containing the XMLTreeView preferred size.

**getTree()**

```
protected javax.swing.JTree getTree()
```

**getXMLTreeModel()**

```
protected oracle.xml.treeviewer.XMLTreeModel getXMLTreeModel()
```

**setXMLDocument(Document)**

```
public void setXMLDocument(org.w3c.dom.Document document)
```

Associates the XMLTreeViewer with a XML document.

**Parameters:**

[doc](#) - The [Document](#) document to display.

**updateUI()**

```
public void updateUI()
```

Forces the XMLTreeView to update/refresh UI.

**Overrides:**

`javax.swing.JPanel.updateUI()` in class `javax.swing.JPanel`

## XMLTreeViewBeanInfo

### Syntax

```
public class XMLTreeViewBeanInfo extends java.beans.SimpleBeanInfo
```

```
java.lang.Object
|
+--java.beans.SimpleBeanInfo
|
+--oracle.xml.treeviewer.XMLTreeViewBeanInfo
```

### All Implemented Interfaces:

```
java.beans.BeanInfo
```

### Constructors

#### XMLTreeViewBeanInfo()

```
public XMLTreeViewBeanInfo()
```

### Methods

#### getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

#### Overrides:

java.beans.SimpleBeanInfo.getIcon(int) in class java.beans.SimpleBeanInfo

#### getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

#### Overrides:

java.beans.SimpleBeanInfo.getPropertyDescriptors() in class java.beans.SimpleBeanInfo



---

## Package oracle.XML.parser.schema

## XMLSchema

```
java.lang.Object
|
+---oracle.xml.parser.schema.XSDConstants
      |
      +---oracle.xml.parser.schema.XSDNode
            |
            +---oracle.xml.parser.schema.XMLSchema
```

```
public class XMLSchema
    extends XSDNode
```

XMLSchema class. Sets top-level XMLSchema document declarations & definitions plus schema location and schema target namespace. XMLSchema objects are created by XSDBuilder as a result of processing XMLSchema documents. They are used by XSDParser for instance XML documents validation and by XSDBuilder as imported schemas.

### Constructor Index

XMLSchema()

XMLSchema constructor.

XMLSchema(int)

XMLSchema constructor.

### Constructors

XMLSchema

```
public XMLSchema() throws XSDEException
```

XMLSchema constructor.

XMLSchema

```
public XMLSchema(int n) throws XSDEException
```

XMLSchema constructor.

### Parameters

n - Initial size of schemanode set.



# XSDBuilder

```
java.lang.Object
|
+---oracle.xml.parser.schema.XSDConstants
|
+---oracle.xml.parser.schema.XSDBuilder
```

```
public class XSDBuilder
    extends XSDConstants
    implements ObjectBuilder
```

Builds an XMLSchema object from XMLSchema document. XMLSchema object is a set of objects (Infoset items) corresponding to top-level schema declarations & definitions. Schema document is 'XML' parsed and converted to a DOM tree. This schema DOM tree is 'Schema' parsed in a following order: (if any) builds a schema object and makes it visible. (if any) is replaced by corresponding DOM tree. Top-level declarations & definitions are registered as a current schema infoset items. Finally, top-level tree elements (infoset items) are 'Schema' parsed. The result XMLSchema object is a set (infoset) of objects (top-level input elements). Object's contents is a tree with nodes corresponding to low-level element/group decls/refs preceded by node/object of type SNode containing cardinality info (min/maxOccurs).

## Constructor Index

```
XSDBuilder()
XSDBuilder constructor
```

## Method Index

`build(InputStream, URL)`  
Build an XMLSchema object

`build(Reader, URL)`  
Build an XMLSchema object

`build(String)`  
Build an XMLSchema object

`build(String, String)`  
Build an XMLSchema object

`build(String, URL)`  
Build an XMLSchema object

`build(URL)`  
Build an XMLSchema object

`build(XMLDocument, URL)`  
Build XMLSchema from XML document

`getObject()`  
Returns the schema object.

`setError(XMLError)`  
Sets XMLError object.

`setLocale(Locale)`  
Sets locale for error reporting.

## Constructors

### **XSDBuilder**

`public XSDBuilder() throws XSDEException`  
XSDBuilder constructor

## Methods

### **setError**

`public void setError(XMLError er)`  
Sets XMLError object.

**Parameters**

er - XMLError object

**setLocale**

```
public void setLocale(Locale locale)
```

Sets locale for error reporting.

**Parameters**

locale - Locale object

**getObject**

```
public Object getObject()
```

Returns the schema object.

**Returns**

XMLSchema object.

**build**

```
public Object build(String sysId) throws Exception
```

Build an XMLSchema object

**Parameters**

sysId - Schema location

**Returns**

Object - XMLSchema

**Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

**build**

```
public Object build(InputStream in, URL baseurl) throws Exception
```

Build an XMLSchema object

**Parameters**

in - Inputstream of Schema

baseurl - URL used to resolve any relative refs.

### **Returns**

Object - XMLSchema

### **Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

### **build**

public Object build(Reader r, URL baseurl) throws Exception

Build an XMLSchema object

### **Parameters**

r - Reader of Schema

baseurl - URL used to resolve any relative refs.

### **Returns**

Object - XMLSchema

### **Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

### **build**

public Object build(URL schemaur) throws Exception

Build an XMLSchema object

### **Parameters**

url - URL of Schema

### **Returns**

Object - XMLSchema

### **Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

### **build**

public Object build(XMLDocument schemaDoc) throws Exception

Build XMLSchema from XML document

**Parameters**

schemaDoc - XMLDocument  
baseurl - URL used to resolve any relative refs.

**Returns**

Object - XMLSchema

**Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

**build**

public Object build(String ns, String sysid) throws Exception  
Build an XMLSchema object

**Parameters**

ns - Schema target namespace used to validate targetNamespace  
sysId - Schema location

**Returns**

Object XMLSchema

**Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

**build**

public Object build(String ns, URL sysid) throws Exception  
Build an XMLSchema object

**Parameters**

ns - Schema target namespace used to validate targetNamespace  
sysId - URL Schema location

**Returns**

Object XMLSchema

**Throws**

An Exception is thrown if Builder fails to build an XMLSchema object.

## XSDException

```
java.lang.Object
|
+--- java.lang.Throwable
      |
      +--- java.lang.Exception
            |
            +--- oracle.xml.parser.schema.XSDException
```

```
public class XSDException
extends Exception
```

Indicates that an exception occurred during XMLSchema validation

### Method Index

```
getMessage()
```

Override getMessage, in order to construct error message from error id, and error params

```
getMessage(XMLError)
```

Get localized message based on the XMLError sent as parameter

### Methods

```
getMessage
```

```
public String getMessage()
```

Override getMessage, in order to construct error message from error id, and error params

### Overrides

getMessage in class Throwable

### getMessage

```
public String getMessage(XMLError err)Get localized message based on the
XMLError sent as parameter
```

### Parameters

err - XMLError class used to get the error message

# Part II

---

## XDK for PL/SQL Packages

This section contains [Chapter 6, "XML Parser for PL/SQL"](#)





---

## XML Parser for PL/SQL

This chapter describes the Extensible Markup Language (XML) Parser for PL/SQL. It has three main sections:

- [PL/SQL Parser APIs](#)
- [Extensible Stylesheet Language \(XSL\) Package Processor APIs](#)
- [W3C Document Object Model \(DOM\) APIs](#)

## PL/SQL Parser APIs

The Extensible Markup Language (XML) describes a class of data objects called XML documents. It partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of the Standard Generalized Markup Language (SGML) By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

A software module called an XML processor is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the application. This PL/SQL implementation of the XML processor (or parser) followed the W3C XML specification (rev. REC-xml-19980210) and included the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

The following is the default behavior for this PL/SQL XML parser:

- A parse tree which can be accessed by DOM APIs is built
- The parser is validating if a DTD is found, otherwise, it is non-validating
- Errors are not recorded unless an error log is specified; however, an application error will be raised if parsing fails

The types and methods described in this document are made available by the PL/SQL package `xmlparser`.

## Types and Functions

### **parse(VARCHAR2)**

Parses XML stored in the given url/file and returns the built DOM Document

### **newParser**

Returns a new parser instance

### **parse(Parser, VARCHAR2)**

Parses xml stored in the given url/file

### **parseBuffer(Parser, VARCHAR2)**

Parses xml stored in the given buffer

### **parseClob(Parser, CLOB)**

Parses xml stored in the given clob

### **parseDTD(Parser, VARCHAR2, VARCHAR2)**

Parses xml stored in the given url/file

### **parseDTDBuffer(Parser, VARCHAR2, VARCHAR2)**

Parses xml stored in the given buffer

### **parseDTDClob(Parser, CLOB, VARCHAR2)**

Parses xml stored in the given clob

### **setBaseDir(Parser, VARCHAR2)**

Sets base directory used to resolve relative urls

### **showWarnings(Parser, BOOLEAN)**

Turn warnings on or off

### **setErrorLog(Parser, VARCHAR2)**

Sets errors to be sent to the specified file

**setPreserveWhitespace(Parser, BOOLEAN)**

Sets white space preserve mode

**setValidationMode(Parser, BOOLEAN)**

Sets validation mode

**getValidationMode(Parser)**

Gets validation mode

**setDoctype(Parser, DOMDocumentType)**

Sets DTD

**getDoctype(Parser)**

Gets DTD Parser

**getDocument(Parser)**

Gets DOM document

**freeParser(Parser)**

Free a Parser object

## Parser Interface Type Description

**TYPE Parser IS RECORD ( ID VARCHAR2(5) );**

## Function Prototypes

### parse

#### **PURPOSE**

Parses xml stored in the given url/file and returns the built DOM Document.

#### **SYNTAX**

```
FUNCTION parse(url VARCHAR2) RETURN DOMDocument;
```

#### **PARAMETERS**

url (IN)- complete path of the url/file to be parsed

#### **RETURNS**

Nothing

#### **COMMENTS**

This is meant to be used when the default parser behavior is acceptable and just a url/file needs to be parsed.

An application error is raised if parsing failed, for some reason.

### newParser

#### **PURPOSE**

Returns a new parser instance

#### **SYNTAX**

```
FUNCTION newParser RETURN Parser;
```

#### **PARAMETERS**

None

## RETURNS

A new parser instance

## COMMENTS

This function must be called before the default behavior of Parser can be changed and if other parse methods need to be used.

## parse

### PURPOSE

Parses xml stored in the given url/file

### SYNTAX

```
PROCEDURE parse(p Parser, url VARCHAR2);
```

### PARAMETERS

p (IN)- parser instance  
url (IN)- complete path of the url/file to be parsed

### RETURNS

Nothing

### COMMENTS

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.

## parseBuffer

### PURPOSE

Parses xml stored in the given buffer

### SYNTAX

```
PROCEDURE parseBuffer(p Parser, doc VARCHAR2);
```

**PARAMETERS**

p (IN)- parser instance  
doc (IN)- xml document buffer to parse

**RETURNS**

Nothing

**COMMENTS**

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.

**parseClob****PURPOSE**

Parses xml stored in the given clob

**SYNTAX**

```
PROCEDURE parseClob(p Parser, doc CLOB);
```

**PARAMETERS**

p (IN)- parser instance  
doc (IN)- xml document clob to parse

**RETURNS**

Nothing

**COMMENTS**

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.

**parseDTD****PURPOSE**

Parses the DTD stored in the given url/file

## SYNTAX

```
PROCEDURE parseDTD(p Parser, url VARCHAR2, root VARCHAR2);
```

## PARAMETERS

p (IN)- parser instance  
url (IN)- complete path of the url/file to be parsed  
root (IN)- name of the root element

## RETURNS

Nothing

## COMMENTS

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.

## parseDTDBuffer

### PURPOSE

Parses the DTD stored in the given buffer

### SYNTAX

```
PROCEDURE parseDTDBuffer(p Parser, dtd VARCHAR2, root VARCHAR2);
```

### PARAMETERS

p (IN)- parser instance  
dtd (IN)- dtd buffer to parse  
root (IN)- name of the root element

### RETURNS

Nothing

### COMMENTS

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.



## parseDTDClob

### PURPOSE

Parses the DTD stored in the given clob

### SYNTAX

```
PROCEDURE parseDTDClob(p Parser, dtd CLOB, root VARCHAR2);
```

### PARAMETERS

p	(IN)-	parser instance
dtd	(IN)-	dtd clob to parse
root	(IN)-	name of the root element

### RETURNS

Nothing

### COMMENTS

Any changes to the default parser behavior should be effected before calling this procedure.

An application error is raised if parsing failed, for some reason.

## setBaseDir

### PURPOSE

Sets base directory used to resolve relative urls

### SYNTAX

```
PROCEDURE setBaseDir(p Parser, dir VARCHAR2);
```

### PARAMETERS

p	(IN)-	parser instance
dir	(IN)-	directory to use as base directory

### RETURNS

Nothing

## COMMENTS

An application error is raised if parsing failed, for some reason.

## showWarnings

### PURPOSE

Turn warnings on or off

### SYNTAX

```
PROCEDURE showWarnings(p Parser, yes BOOLEAN);
```

### PARAMETERS

p (IN)- parser instance  
yes (IN)- mode to set: TRUE - show warnings, FALSE - don't show warnings

### RETURNS

Nothing

## setErrorLog

### PURPOSE

Sets errors to be sent to the specified file

### SYNTAX

```
PROCEDURE setErrorLog(p Parser, fileName VARCHAR2);
```

### PARAMETERS

p (IN)- parser instance  
fileName (IN)- complete path of the file to use as the error log

### RETURNS

Nothing

## setPreserveWhitespace

### PURPOSE

Sets whitespace preserving mode

**SYNTAX**

```
PROCEDURE setPreserveWhitespace(p Parser, yes BOOLEAN);
```

**PARAMETERS**

```
p          (IN)- parser instance  
yes       (IN)- mode to set: TRUE - preserve, FALSE - don't preserve
```

**RETURNS**

Nothing

**setValidationMode****PURPOSE**

Sets validation mode

**SYNTAX**

```
PROCEDURE setValidationMode(p Parser, yes BOOLEAN);
```

**PARAMETERS**

```
p          (IN)- parser instance  
yes       (IN)- mode to set: TRUE - validating, FALSE - non valid
```

**RETURNS**

Nothing

**getValidationMode****PURPOSE**

Gets validation mode

**SYNTAX**

```
FUNCTION getValidationMode(p Parser) RETURN BOOLEAN;
```

**PARAMETERS**

```
p          (IN)- parser instance
```

## **RETURNS**

The validation mode: TRUE - validating, FALSE - non valid

## **setDoctype**

### **PURPOSE**

Sets a DTD to be used by the parser for validation

### **SYNTAX**

```
PROCEDURE setDoctype(p Parser, dtd DOMDocumentType);
```

### **PARAMETERS**

p (IN)- parser instance  
dtd (IN)- DTD to set

### **RETURNS**

Nothing

## **getDoctype**

### **PURPOSE**

Gets DTD - MUST be called only after a DTD is parsed

### **SYNTAX**

```
FUNCTION getDoctype(p Parser) RETURN DOMDocumentType;
```

### **PARAMETERS**

p (IN)- parser instance

### **RETURNS**

The parsed DTD

## **getDocument**

### **PURPOSE**

Gets DOM Document built by the parser - MUST be called only after a document is parsed

**SYNTAX**

```
FUNCTION getDocument(p Parser) RETURN DOMDocument;
```

**PARAMETERS**

p (IN)- parser instance

**RETURNS**

The root of the DOM tree

**freeParser****PURPOSE**

Free a parser object

**SYNTAX**

```
PROCEDURE freeParser(p Parser)
```

**PARAMETERS**

p (IN)- parser instance

## Extensible Stylesheet Language (XSL) Package Processor APIs

The Extensible Stylesheet Language Transformation (XSLT), describes rules for transforming a source tree into a result tree. A transformation expressed in XSLT is called a stylesheet. The transformation specified is achieved by associating patterns with templates defined in the stylesheet. A template is instantiated to create part of the result tree. This PL/SQL implementation of the XSL processor followed the W3C XSLT working draft (rev WD-xslt-19990813) and included the required behavior of an XSL processor in terms of how it must read XSLT stylesheets and the transformation it must effect.

The following is the default behavior for this PL/SQL XML parser:

- A result tree which can be accessed by DOM APIs is built
- Errors are not recorded unless an error log is specified; however, an application error will be raised if parsing fails

The types and methods described in this document are made available by the PL/SQL package `xslprocessor`.

## Functions

### newProcessor

#### **PURPOSE**

Returns a new processor instance

#### **SYNTAX**

```
FUNCTION newProcessor RETURN Processor;
```

#### **PARAMETERS**

None

#### **RETURNS**

A new processor instance

#### **COMMENTS**

This function must be called before the default behavior of Processor can be changed and if other processor methods need to be used.

### processXSL

#### **PURPOSE**

Transforms input XML document using given DOMDocument and stylesheet

#### **SYNTAX**

```
PROCEDURE processXSL(p Processor, ss Stylesheet, xmlDoc DOMDocument);
```

#### **PARAMETERS**

p (IN)- processor instance  
ss (IN)- stylesheet instance  
xmlDoc (IN)- xml document to be transformed

#### **RETURNS**

Nothing

## COMMENTS

Any changes to the default processor behavior should be effected before calling this procedure.

An application error is raised if processing failed, for some reason.

## processXSL

### PURPOSE

Transforms input XML document using given DOMDocumentFragment and stylesheet

### SYNTAX

```
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldoc DOMDocumentFragment);
```

### PARAMETERS

p (IN)- processor instance  
ss (IN)- stylesheet instance  
xmldoc (IN)- xml document fragment to be transformed

### RETURNS

Nothing

## COMMENTS

Any changes to the default processor behavior should be effected before calling this procedure.

An application error is raised if processing failed, for some reason.

## showWarnings

### PURPOSE

Turn warnings on or off

### SYNTAX

```
PROCEDURE showWarnings(p Processor, yes BOOLEAN);
```

### PARAMETERS

p (IN)- processor instance  
yes (IN)- mode to set: TRUE - show warnings, FALSE - don't show warnings

## RETURNS

Nothing

## setErrorLog

### PURPOSE

Sets errors to be sent to the specified file

### SYNTAX

```
PROCEDURE setErrorLog(p Processor, fileName VARCHAR2);
```

### PARAMETERS

p (IN)- processor instance  
fileName (IN)- complete path of the file to use as the error log

## RETURNS

Nothing

## newStylesheet

### PURPOSE

Create a new stylesheet using the given input and reference URLs

### SYNTAX

```
FUNCTION newStylesheet(inp VARCHAR2, ref VARCHAR2) RETURN Stylesheet;
```

### PARAMETERS

inp (IN)- input url to use for construction  
ref (IN)- reference url

## RETURNS

A new stylesheet instance

## transformNode

### PURPOSE

Transforms a node in a DOM tree using the given stylesheet



**SYNTAX**

```
FUNCTION transformNode(n DOMNode, ss Stylesheet) RETURN DOMDocumentFragment;
```

**PARAMETERS**

```
n          (IN)- DOM Node to transform
ss         (IN)- stylesheet to use
```

**RETURNS**

Result of the transformation

**selectNodes****PURPOSE**

Selects nodes from a DOM tree which match the given pattern

**SYNTAX**

```
FUNCTION selectNodes(n DOMNode, pattern VARCHAR2) RETURN DOMNodeList;
```

**PARAMETERS**

```
n          (IN)- DOM Node to transform
pattern    (IN)- pattern to use
```

**RETURNS**

Result of the selection

**selectSingleNode****PURPOSE**

Selects the first node from the tree that matches the given pattern

**SYNTAX**

```
FUNCTION selectSingleNode(n DOMNode, pattern VARCHAR2) RETURN DOMNode;
```

**PARAMETERS**

```
n          (IN)- DOM Node to transform
pattern    (IN)- pattern to use
```

## RETURNS

Result of the selection

## valueOf

## PURPOSE

Retrieves the value of the first node from the tree that matches the given pattern

## SYNTAX

```
FUNCTION valueOf(n DOMNode, pattern VARCHAR2) RETURN VARCHAR2;
```

## PARAMETERS

n (IN)- DOM Node to transform  
pattern (IN)- pattern to use

## RETURNS

Result of the selection

## setParam

## PURPOSE

Sets a top level paramter in the stylesheet

## SYNTAX

```
PROCEDURE setParam(ss Stylesheet, name VARCHAR2, value VARCHAR2)
```

## PARAMETERS

ss (IN)- stylesheet  
name (IN)- name of the param  
value (IN)- value of the param

## removeParam

## PURPOSE

Removes a top level stylesheet parameter

**SYNTAX**

```
PROCEDURE removeParam(ss Stylesheet, name VARCHAR2)
```

**PARAMETERS**

```
ss          (IN)- Stylesheet  
name       (IN)- name of the parameter
```

**resetParams****PURPOSE**

Resets the top-level stylesheet parameters

**SYNTAX**

```
PROCEDURE resetParams(ss Stylesheet)
```

**PARAMETERS**

```
ss          (IN)- Stylesheet
```

**freeStylesheet****PURPOSE**

Frees a Stylesheet object

**SYNTAX**

```
PROCEDURE freestylesheet(ss Stylesheet)
```

**PARAMETERS**

```
ss          (IN)- Stylesheet
```

**freeProcessor****PURPOSE**

Frees a Processor object

**SYNTAX**

```
PROCEDURE freeProcessor(p Processor)
```

## PARAMETERS

p (IN)- Processor

## W3C Document Object Model (DOM) APIs

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense. XML is increasingly being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions. In particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

One important objective of the W3C specification for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The DOM is designed to be used with any programming language. Since the DOM standard is object-oriented, for this PL/SQL adaptation, some changes had to be made:

- Various DOM interfaces such as Node, Element, etc. have equivalent PL/SQL types DOMNode, DOMELEMENT, etc. respectively.
- Various DOMException codes such as WRONG\_DOCUMENT\_ERR, HIERARCHY\_REQUEST\_ERR, etc. have similarly named PL/SQL exceptions
- Various DOM Node type codes such as ELEMENT\_NODE, ATTRIBUTE\_NODE, etc. have similarly named PL/SQL constants
- Methods defined on a DOM type become functions or procedures that accept it as a parameter. For example, to perform appendChild on a DOM Node n, the following PL/SQL function is provided:

### FUNCTION

```
appendChild(n DOMNode, newChild IN DOMNode)
```

### RETURN

```
DOMNode;
```

and to perform setAttribute on a DOM Element elem, the following PL/SQL procedure is provided:

## PROCEDURE

```
setAttribute(elem DOMELEMENT, name IN VARCHAR2,  
             value IN VARCHAR2);
```

DOM defines an inheritance hierarchy. For example, Document, Element, and Attr are defined to be subtypes of Node. Thus, a method defined in the Node interface should be available in these as well. Since, such inheritance is not directly possible in PL/SQL, the makeNode functions need to be invoked on different DOM types to convert these into a DOMNode. The appropriate functions or procedures that accept DOMNodes can then be called to operate on these types. If, subsequently, type specific functionality is desired, the DOMNode can be converted back into the type by using the makeXXX functions, where DOMXXX is the DOM type desired

The implementation of this PL/SQL DOM interface followed the DOM standard of revision REC-DOM-Level-1-19981001. The types and methods described in this document are made available by the PL/SQL package `xml.dom`.

# Types

## DOM Node Types

ELEMENT\_NODE  
ATTRIBUTE\_NODE  
TEXT\_NODE  
CDATA\_SECTION\_NODE  
ENTITY\_REFERENCE\_NODE  
ENTITY\_NODE  
PROCESSING\_INSTRUCTION\_NODE  
COMMENT\_NODE  
DOCUMENT\_NODE  
DOCUMENT\_TYPE\_NODE  
DOCUMENT\_FRAGMENT\_NODE  
NOTATION\_NODE

## DOM Exception Types

INDEX\_SIZE\_ERR  
DOMSTRING\_SIZE\_ERR  
HIERARCHY\_REQUEST\_ERR  
WRONG\_DOCUMENT\_ERR  
INVALID\_CHARACTER\_ERR  
NO\_DATA\_ALLOWED\_ERR  
NO\_MODIFICATION\_ALLOWED\_ERR  
NOT\_FOUND\_ERR  
NOT\_SUPPORTED\_ERR  
INUSE\_ATTRIBUTE\_ERR

## DOM Interface Types

DOMNode

DOMNamedNodeMap

DOMNodeList

DOMAttr

DOMCDATASection

DOMCharacterData

DOMComment

DOMDocumentFragment

DOMElement

DOMEntity

DOMEntityReference

DOMNotation

DOMProcessingInstruction

DOMText

DOMImplementation

DOMDocumentType

DOMDocument



## Methods

### Node Methods

**FUNCTION**

isNull(n DOMNode)

**RETURN**

BOOLEAN;

**FUNCTION**

makeAttr(n DOMNode)

**RETURN**

DOMAttr;

**FUNCTION**

makeCDataSection(n DOMNode)

**RETURN**

DOMCDataSection;

**FUNCTION**

makeCharacterData(n DOMNode)

**RETURN**

DOMCharacterData;

**FUNCTION**

makeComment(n DOMNode)

**RETURN**

DOMComment;

**FUNCTION**

makeDocumentFragment(n DOMNode)

**RETURN**

DOMDocumentFragment;

**FUNCTION**

makeDocumentType(n DOMNode)

**RETURN**

DOMDocumentType;

**FUNCTION**

makeElement(n DOMNode)

**RETURN**

DOMElement;

**FUNCTION**

makeEntity(n DOMNode)

**RETURN**

DOMEntity;

**FUNCTION**

makeEntityReference(n DOMNode)

**RETURN**

DOMEntityReference;

**FUNCTION**

makeNotation(n DOMNode)

**RETURN**

DOMNotation;

**FUNCTION**

makeProcessingInstruction(n DOMNode)

**RETURN**

DOMProcessingInstruction;

makeText(n DOMNode) RETURN DOMText;

**FUNCTION**

makeDocument(n DOMNode)

**RETURN**

DOMDocument;

**PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2);

**PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);

**PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB);

**PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);

**FUNCTION**

getNodeName(n DOMNode)

**RETURN**

VARCHAR2;

**FUNCTION**

getNodeValue(n DOMNode)

**RETURN**

VARCHAR2;

**PROCEDURE**

setNodeValue(n DOMNode, nodeValue IN VARCHAR2);

**FUNCTION**

getNodeType(n DOMNode)

**RETURN**

NUMBER;

**FUNCTION**

getParentNode(n DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

getChildNodes(n DOMNode)

**RETURN**

DOMNodeList;

**FUNCTION**

getFirstChild(n DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

getLastChild(n DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

getPreviousSibling(n DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

getNextSibling(n DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

getAttributes(n DOMNode)

**RETURN**

DOMNamedNodeMap;

**FUNCTION**

getOwnerDocument(n DOMNode)

**RETURN**

DOMDocument;

**FUNCTION**

insertBefore(n DOMNode, newChild IN DOMNode, refChild IN DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

replaceChild(n DOMNode, newChild IN DOMNode, oldChild IN DOMNode)&gt;

**RETURN**

DOMNode;

**FUNCTION**

removeChild(n DOMNode, oldChild IN DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

appendChild(n DOMNode, newChild IN DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

hasChildNodes(n DOMNode)

**RETURN**

BOOLEAN;

**FUNCTION**

cloneNode(n DOMNode, deep boolean)

**RETURN**

DOMNode;

## Named Node Map Methods

**FUNCTION**

isNull(nnm DOMNamedNodeMap)

**RETURN**

BOOLEAN;

**FUNCTION**

getNamedItem(nnm DOMNamedNodeMap, name IN VARCHAR2)

**RETURN**

DOMNode;

**FUNCTION**

setNamedItem(nnm DOMNamedNodeMap, arg IN DOMNode)

**RETURN**

DOMNode;

**FUNCTION**

removeNamedItem(nnm DOMNamedNodeMap, name IN VARCHAR2)

**RETURN**

DOMNode;

**FUNCTION**

item(nnm DOMNamedNodeMap, idx IN NUMBER)

**RETURN**

DOMNode;

**FUNCTION**

getLength(nnm DOMNamedNodeMap)

**RETURN**  
NUMBER;



## Node List Methods

**FUNCTION**

isNull(nl DOMNodeList)

**RETURN**

BOOLEAN;

**FUNCTION**

item(nl DOMNodeList, idx IN NUMBER)

**RETURN**

DOMNode;

**FUNCTION**

getLength(nl DOMNodeList)

**RETURN**

NUMBER;

## Attr Methods

**FUNCTION**

isNull(a DOMAttr)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(a DOMAttr)

**RETURN**

DOMNode;

**FUNCTION**

getQualifiedName(a DOMAttr)

**RETURN**

VARCHAR2;

**FUNCTION**

getNamespace(a DOMAttr)

**RETURN**

VARCHAR2;

**FUNCTION**

getLocalName(a DOMAttr)

**RETURN**

VARCHAR2;

**FUNCTION**

getExpandedName(a DOMAttr)

**RETURN**

VARCHAR2;

**FUNCTION**

getName(a DOMAttr)

**RETURN**

VARCHAR2;

**FUNCTION**

getSpecified(a DOMAttr)

**RETURN**

BOOLEAN;

**FUNCTION**

getValue(a DOMAttr)

**RETURN**

VARCHAR2;

**PROCEDURE**

setValue(a DOMAttr, value IN VARCHAR2);

## C Data Section Methods

### **FUNCTION**

isNull(cds DOMCDataSection)

### **RETURN**

BOOLEAN;

### **FUNCTION**

makeNode(cds DOMCDataSection)

### **RETURN**

DOMNode;

## Character Data Methods

**FUNCTION**

isNull(cd DOMCharacterData)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(cd DOMCharacterData)

**RETURN**

DOMNode;

**FUNCTION**

getData(cd DOMCharacterData)

**RETURN**

VARCHAR2;

**PROCEDURE**

setData(cd DOMCharacterData, data IN VARCHAR2);

**FUNCTION**

getLength(cd DOMCharacterData)

**RETURN**

NUMBER;

**FUNCTION**

substringData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER)

**RETURN**

VARCHAR2;

**PROCEDURE**

appendData(cd DOMCharacterData, arg IN VARCHAR2);

**PROCEDURE**

insertData(cd DOMCharacterData, offset IN NUMBER, arg IN VARCHAR2);

**PROCEDURE**

deleteData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER);

**PROCEDURE**

replaceData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER, arg IN VARCHAR2);

## Comment Methods

**FUNCTION**

isNull(com DOMComment)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(com DOMComment)

**RETURN**

DOMNode;

## DOM Implementation Methods

**FUNCTION**

isNull(di DOMImplementation)

**RETURN**

BOOLEAN;

**FUNCTION**

hasFeature(di DOMImplementation, feature IN VARCHAR2, version IN VARCHAR2)

**RETURN**

BOOLEAN;

**Document Fragment Methods****FUNCTION**

isNull(df DOMDocumentFragment)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(df DOMDocumentFragment)

**RETURN**

DOMNode;

**Document Type Methods****FUNCTION**

isNull(dt DOMDocumentType)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(dt DOMDocumentType)

**RETURN**

DOMNode;

**FUNCTION**

findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN)

**RETURN**

DOMEntity;

**FUNCTION**

findNotation(dt DOMDocumentType, name VARCHAR2)

**RETURN**

DOMNotation;

**FUNCTION**

getPublicId(dt DOMDocumentType)

**RETURN**

VARCHAR2;

**FUNCTION**

getSystemId(dt DOMDocumentType)

**RETURN**

VARCHAR2;

**PROCEDURE**

writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);

**PROCEDURE**

writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);

**PROCEDURE**

writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);

**PROCEDURE**

writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2);



**PROCEDURE**

writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2,  
charset VARCHAR2);

**PROCEDURE**

writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset  
VARCHAR2);

**FUNCTION**

getName(dt DOMDocumentType)

**RETURN**

VARCHAR2;

**FUNCTION**

getEntities(dt DOMDocumentType)

**RETURN**

DOMNamedNodeMap;

**FUNCTION**

getNotations(dt DOMDocumentType)

**RETURN**

DOMNamedNodeMap;

## Element Methods

**FUNCTION**

isNull(elem DOMElement)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(elem DOMElement)

**RETURN**

DOMNode;

**FUNCTION**

getQualifiedName(elem DOMElement)

**RETURN**

VARCHAR2;

**FUNCTION**

getNamespace(elem DOMElement)

**RETURN**

VARCHAR2;

**FUNCTION**

getLocalName(elem DOMElement)

**RETURN**

VARCHAR2;

**FUNCTION**

getExpandedName(elem DOMElement)

**RETURN**

VARCHAR2;

**FUNCTION**

getChildrenByTagName(elem DOMELEMENT, name IN VARCHAR2)

**RETURN**

DOMNodeList;

**FUNCTION**getElementsByTagName(elem DOMELEMENT, name IN VARCHAR2, ns  
VARCHAR2)**RETURN**

DOMNodeList;

**FUNCTION**getChildrenByTagName(elem DOMELEMENT, name IN VARCHAR2, ns  
VARCHAR2)**RETURN**

DOMNodeList;

**FUNCTION**

resolveNamespacePrefix(elem DOMELEMENT, prefix VARCHAR2)

**RETURN**

VARCHAR2;

**FUNCTION**

getTagName(elem DOMELEMENT)

**RETURN**

VARCHAR2;

**FUNCTION**

getAttribute(elem DOMELEMENT, name IN VARCHAR2)

**RETURN**

VARCHAR2;

**PROCEDURE**

setAttribute(elem DOMELEMENT, name IN VARCHAR2, value IN VARCHAR2);

**PROCEDURE**

removeAttribute(elem DOMELEMENT, name IN VARCHAR2);

**FUNCTION**

getAttributeNode(elem DOMELEMENT, name IN VARCHAR2)

**RETURN**

DOMAttr;

**FUNCTION**

setAttributeNode(elem DOMELEMENT, newAttr IN DOMAttr)

**RETURN**

DOMAttr;

**FUNCTION**

removeAttributeNode(elem DOMELEMENT, oldAttr IN DOMAttr)

**RETURN**

DOMAttr;

**FUNCTION**

getElementsByTagName(elem DOMELEMENT, name IN VARCHAR2)

**RETURN**

DOMNodeList;

**PROCEDURE**

```
normalize(elem DOMElement);
```

## Entity Methods

**FUNCTION**

isNull(ent DOMEntity)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(ent DOMEntity)

**RETURN**

DOMNode;

**FUNCTION**

getPublicId(ent DOMEntity)

**RETURN**

VARCHAR2;

**FUNCTION**

getSystemId(ent DOMEntity)

**RETURN**

VARCHAR2;

**FUNCTION**

getNotationName(ent DOMEntity)

**RETURN**

VARCHAR2;

## Entity Reference Methods

**FUNCTION**

isNull(eref DOMEntityReference)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(eref DOMEntityReference)

**RETURN**

DOMNode;

## Notation Methods

### **FUNCTION**

isNull(n DOMNotation)

### **RETURN**

BOOLEAN;

### **FUNCTION**

makeNode(n DOMNotation)

### **RETURN**

DOMNode;

### **FUNCTION**

getPublicId(n DOMNotation)

### **RETURN**

VARCHAR2;

### **FUNCTION**

getSystemId(n DOMNotation)

### **RETURN**

VARCHAR2;



## Processing Instruction Methods

**FUNCTION**

isNull(pi DOMProcessingInstruction)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(pi DOMProcessingInstruction)

**RETURN**

DOMNode;

**FUNCTION**

getData(pi DOMProcessingInstruction)

**RETURN**

VARCHAR2;

**FUNCTION**

getTarget(pi DOMProcessingInstruction)

**RETURN**

VARCHAR2;

**PROCEDURE**

setData(pi DOMProcessingInstruction, data IN VARCHAR2);

## Text Methods

**FUNCTION**

isNull(t DOMText)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(t DOMText)

**RETURN**

DOMNode;

**FUNCTION**

splitText(t DOMText, offset IN NUMBER)

**RETURN**

DOMText;

## Document Methods

**FUNCTION**

isNull(doc DOMDocument)

**RETURN**

BOOLEAN;

**FUNCTION**

makeNode(doc DOMDocument)

**RETURN**

DOMNode;

**FUNCTION**

newDOMDocument

**RETURN**

DOMDocument;

**FUNCTION**

getVersion(doc DOMDocument)

**RETURN**

VARCHAR2;

**PROCEDURE**

setVersion(doc DOMDocument, version VARCHAR2);

**FUNCTION**

getCharset(doc DOMDocument)

**RETURN**

VARCHAR2;

**PROCEDURE**

setCharset(doc DOMDocument, charset VARCHAR2);

**FUNCTION**

getStandalone(doc DOMDocument)

**RETURN**

VARCHAR2;

**PROCEDURE**

setStandalone(doc DOMDocument, value VARCHAR2);

**PROCEDURE**

writeToFile(doc DOMDocument, fileName VARCHAR2);

**PROCEDURE**

writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);

**PROCEDURE**

writeToClob(doc DOMDocument, cl IN OUT CLOB);

**PROCEDURE**

writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);

**PROCEDURE**

writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2);

**PROCEDURE**

writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);

**PROCEDURE**

writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);

**PROCEDURE**

writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);

**PROCEDURE**

writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);

**FUNCTION**

getDoctype(doc DOMDocument)

**RETURN**

DOMDocumentType;

**FUNCTION**

getImplementation(doc DOMDocument)

**RETURN**

DOMImplementation;

**FUNCTION**

getDocumentElement(doc DOMDocument)

**RETURN**

DOMElement;

**FUNCTION**

createElement(doc DOMDocument, tagName IN VARCHAR2)

**RETURN**

DOMElement;

**FUNCTION**

createDocumentFragment(doc DOMDocument)

**RETURN**

DOMDocumentFragment;

**FUNCTION**

createTextNode(doc DOMDocument, data IN VARCHAR2)

**RETURN**

DOMText;

**FUNCTION**

createComment(doc DOMDocument, data IN VARCHAR2)

**RETURN**

DOMComment;

**FUNCTION**

createCDATASection(doc DOMDocument, data IN VARCHAR2)

**RETURN**

DOMCDATASection;

**FUNCTION**

createProcessingInstruction(doc DOMDocument, target IN VARCHAR2,  
data IN VARCHAR2)

**RETURN**

DOMProcessingInstruction;

**FUNCTION**

createAttribute(doc DOMDocument, name IN VARCHAR2)

**RETURN**

DOMAttr;

**FUNCTION**

createEntityReference(doc DOMDocument, name IN VARCHAR2)

**RETURN**

DOMEntityReference;

**FUNCTION**

getElementsByTagName(doc DOMDocument, tagname IN VARCHAR2)

**RETURN**

DOMNodeList;

## Method Prototypes

### Node Methods

#### **FUNCTION**

`isNull(n DOMNode)`

#### **RETURN**

BOOLEAN;

#### **PURPOSE**

Checks if the given DOMNode is null

#### **SYNTAX**

```
FUNCTION isNull(n DOMNode) RETURN BOOLEAN;
```

#### **PARAMETERS**

n (IN)- DOMNode to check

#### **RETURNS**

Whether given DOMNode is null: TRUE - is null, FALSE - is not null

#### **FUNCTION**

`makeAttr(n DOMNode) RETURN DOMAttr;`

#### **PURPOSE**

Casts given DOMNode to a DOMAttr

#### **SYNTAX**

```
FUNCTION makeAttr(n DOMNode) RETURN DOMAttr;
```

#### **PARAMETERS**

n (IN)- DOMNode to cast

#### **RETURNS**

The DOMAttr



**FUNCTION**

**makeCDATASection(n DOMNode)** RETURN DOMCDATASection;

**PURPOSE**

Casts given DOMNode to a DOMCDATASection

**SYNTAX**

```
FUNCTION makeCDATASection(n DOMNode) RETURN DOMCDATASection;
```

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMCDATASection

**FUNCTION**

**makeCharacterData(n DOMNode)** RETURN DOMCharacterData;

**PURPOSE**

Casts given DOMNode to a DOMCharacterData

**SYNTAX**

```
FUNCTION makeCharacterData(n DOMNode) RETURN DOMCharacterData;
```

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMCharacterData

**FUNCTION**

**makeComment(n DOMNode)** RETURN DOMComment;

**PURPOSE**

Casts given DOMNode to a DOMComment

## SYNTAX

```
FUNCTION makeComment(n DOMNode) RETURN DOMComment;
```

## PARAMETERS

n (IN)- DOMNode to cast

## RETURNS

The DOMComment

## FUNCTION

```
makeDocumentFragment(n DOMNode) RETURN DOMDocumentFragment;
```

## PURPOSE

Casts given DOMNode to a DOMDocumentFragment

## SYNTAX

```
FUNCTION makeDocumentFragment(n DOMNode) RETURN DOMDocumentFragment;
```

## PARAMETERS

n (IN)- DOMNode to cast

## RETURNS

The DOMDocumentFragment

```
FUNCTION makeDocumentType(n DOMNode) RETURN DOMDocumentType;
```

## PURPOSE

Casts given DOMNode to a DOMDocumentType

## SYNTAX

```
FUNCTION makeDocumentType(n DOMNode) RETURN DOMDocumentType;
```

## PARAMETERS

n (IN)- DOMNode to cast

## RETURNS

The DOMDocumentType

**FUNCTION**

`makeElement(n DOMNode)`

**RETURNS**

DOMElement;

**PURPOSE**

Casts given DOMNode to a DOMElement

**SYNTAX**

```
FUNCTION makeElement(n DOMNode) RETURN DOMElement;
```

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMElement

```
FUNCTION makeEntity(n DOMNode) RETURN DOMEntity;
```

**PURPOSE**

Casts given DOMNode to a DOMEntity

**SYNTAX**

```
FUNCTION makeEntity(n DOMNode) RETURN DOMEntity;
```

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMEntity

```
FUNCTION makeEntityReference(n DOMNode) RETURN DOMEntityReference;
```

**PURPOSE**

Casts given DOMNode to a DOMEntityReference

## SYNTAX

```
FUNCTION makeEntityReference(n DOMNode) RETURN DOMEntityReference;
```

## PARAMETERS

n (IN)- DOMNode to cast

## RETURNS

The DOMEntityReference

## FUNCTION

```
makeNotation(n DOMNode)
```

## RETURN

DOMNotation;

## PURPOSE

Casts given DOMNode to a DOMNotation

## SYNTAX

## FUNCTION

```
makeNotation(n DOMNode)
```

## RETURN

DOMNotation;

## PARAMETERS

n (IN)- DOMNode to cast

## RETURNS

The DOMNotation

## FUNCTION

```
makeProcessingInstruction(n DOMNode)
```

## RETURN

DOMProcessingInstruction;

**PURPOSE**

Casts given DOMNode to a DOMProcessingInstruction

**SYNTAX****FUNCTION**

makeProcessingInstruction(n DOMNode)

**RETURN**

DOMProcessingInstruction;

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMProcessingInstruction

**FUNCTION**

**makeText(n DOMNode)**

**RETURN**

DOMText;

**PURPOSE**

Casts given DOMNode to a DOMText

**SYNTAX****FUNCTION**

makeText(n DOMNode)

**RETURN**

DOMText ;

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMText

**FUNCTION**

makeDocument(n DOMNode)

**RETURN**

DOMDocument;

**PURPOSE**

Casts given DOMNode to a DOMDocument

**SYNTAX**

**FUNCTION**

makeDocument (n DOMNode)

**RETURN**

DOMDocument ;

**PARAMETERS**

n (IN)- DOMNode to cast

**RETURNS**

The DOMDocument

**PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2);

**PURPOSE**

Writes XML node to specified file using the database character set

**SYNTAX**

**PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2);

**PARAMETERS**

n (IN)- DOMNode  
fileName (IN)- File to write to

**RETURNS**

Nothing

**PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);

**PURPOSE**

Writes XML node to specified buffer using the database character set

**SYNTAX****PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);

**PARAMETERS**

n (IN)- DOMNode  
buffer (OUT)- buffer to write to

**RETURNS**

Nothing

**PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB);

**PURPOSE**

Writes XML node to specified clob using the database character set

**SYNTAX****PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB);

**PARAMETERS**

n (IN)- DOMNode

clob (OUT)- CLOB to write to

### **RETURNS**

Nothing

### **PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2);

### **PURPOSE**

Writes XML node to specified file using the given character set

### **SYNTAX**

### **PROCEDURE**

writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2);

### **PARAMETERS**

n (IN)- DOMNode  
fileName (IN)- File to write to  
charset (IN)- Character set

### **RETURNS**

Nothing

### **PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);

### **PURPOSE**

Writes XML node to specified buffer using the given character set

### **SYNTAX**

### **PROCEDURE**

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);



**PARAMETERS**

n (IN)- DOMNode  
buffer (OUT)- buffer to write to  
charset (IN)- Character set

**RETURNS**

Nothing

**PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);

**PURPOSE**

Writes XML node to specified clob using the given character set

**SYNTAX****PROCEDURE**

writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);

**PARAMETERS**

n (IN)- DOMNode  
cl (OUT)- CLOB to write to  
charset (IN)- Character set

**RETURNS**

Nothing

**Named Node Map Methods****FUNCTION**

isNull(nnm DOMNamedNodeMap)

**RETURN**

BOOLEAN;

**PURPOSE**

Checks if the given DOM NamedNodeMap is null

## SYNTAX

## FUNCTION

`isNull(nrm DOMNamedNodeMap)`

## RETURN

BOOLEAN;

## PARAMETERS

`nrm` (IN)- DOMNamedNodeMap to check

## RETURNS

Whether given DOM NamedNodeMap is null: TRUE - is null, FALSE - is not null

## Node List Methods

## FUNCTION

`isNull(nl DOMNodeList)`

## RETURN

BOOLEAN;

## PURPOSE

Checks if the given DOM NodeList is null

## SYNTAX

`FUNCTION isNull(nl DOMNodeList) RETURN BOOLEAN;`

## PARAMETERS

`nl` (IN)- DOMNodeList to check

## RETURNS

Whether given DOM NodeList is null: TRUE - is null, FALSE - is not null

## Attr Methods

```
FUNCTION isNull(a DOMAttr) RETURN BOOLEAN;
```

### PURPOSE

Checks if the given DOM Attr is null

### SYNTAX

```
FUNCTION isNull(a DOMAttr) RETURN BOOLEAN;
```

### PARAMETERS

a (IN)- DOMAttr to check

### RETURNS

Whether given DOM Attr is null: TRUE - is null, FALSE - is not null

```
FUNCTION makeNode(a DOMAttr) RETURN DOMNode;
```

### PURPOSE

Casts given DOMAttr to a DOMNode

### SYNTAX

```
FUNCTION makeNode(a DOMAttr) RETURN DOMNode;
```

### PARAMETERS

a (IN)- DOMNode to cast

### RETURNS

The DOMNode

```
FUNCTION getQualifiedName(a DOMAttr) RETURN VARCHAR2;
```

### PURPOSE

Returns the qualified name of the DOMAttr

### SYNTAX

```
FUNCTION getQualifiedName(a DOMAttr) RETURN VARCHAR2;
```

## PARAMETERS

a (IN)- DOMAttr

## RETURNS

The qualified name

FUNCTION **getNamespace(a DOMAttr)** RETURN VARCHAR2;

## PURPOSE

Returns the namespace of the DOMAttr

## SYNTAX

FUNCTION getNamespace(a DOMAttr) RETURN VARCHAR2;

## PARAMETERS

a (IN)- DOMAttr

## RETURNS

The namespace

FUNCTION **getLocalName(a DOMAttr)** RETURN VARCHAR2;

## PURPOSE

Returns the local name of the DOMAttr

## SYNTAX

FUNCTION getLocalName(a DOMAttr) RETURN VARCHAR2;

## PARAMETERS

a (IN)- DOMAttr

## RETURNS

The local name

FUNCTION **getExpandedName(a DOMAttr)** RETURN VARCHAR2;

## PURPOSE

Returns the expanded name of the DOMAttr

**SYNTAX**

```
FUNCTION getExpandedName(a DOMAttr) RETURN VARCHAR2;
```

**PARAMETERS**

a (IN)- DOMAttr

**RETURNS**

The expanded name

**C Data Section Methods****FUNCTION**

```
isNull(cds DOMCDataSection)
```

**RETURN**

BOOLEAN;

**PURPOSE**

Checks if the given DOM CDataSection is null

**SYNTAX**

```
FUNCTION isNull(cds DOMCDataSection) RETURN BOOLEAN;
```

**PARAMETERS**

cds (IN)- DOMCDataSection to check

**RETURNS**

Whether given DOM CDataSection is null: TRUE - is null, FALSE - is not null

**FUNCTION**

```
makeNode(cds DOMCDataSection)
```

**RETURN**

DOMNode;

### **PURPOSE**

Casts given DOMCDataSection to a DOMNode

### **SYNTAX**

```
FUNCTION makeNode(cds DOMCDataSection) RETURN DOMNode;
```

### **PARAMETERS**

cds (IN)- DOMCDataSection to cast

### **RETURNS**

The DOMNode

## **Character Data Methods**

### **FUNCTION**

```
isNull(cd DOMCharacterData)
```

### **RETURN**

BOOLEAN;

### **PURPOSE**

Checks if the given DOM CharacterData is null

### **SYNTAX**

```
FUNCTION isNull(cd DOMCharacterData) RETURN BOOLEAN;
```

### **PARAMETERS**

cd (IN)- DOMCharacterData to check

### **RETURNS**

Whether given DOM CharacterData is null : TRUE - is null, FALSE - is not null

### **FUNCTION**

```
makeNode(cd DOMCharacterData)
```

**RETURN**

DOMNode;

**PURPOSE**

Casts given DOMCharacterData to a DOMNode

**SYNTAX**

```
FUNCTION makeNode(cd DOMCharacterData) RETURN DOMNode;
```

**PARAMETERS**

cd (IN)- DOMCharacterData to cast

**RETURNS**

The DOMNode

## Comment Methods

**FUNCTION**

```
isNull(com DOMComment)
```

**RETURN**

BOOLEAN;

**PURPOSE**

Checks if the given DOM Comment is null

**SYNTAX**

```
FUNCTION isNull(com DOMComment) RETURN BOOLEAN;
```

**PARAMETERS**

com (IN)- DOMComment to check

**RETURNS**

Whether given DOM Comment is null: TRUE - is null, FALSE - is not null

**FUNCTION**

`makeNode(com DOMComment)`

**RETURN**

DOMNode;

**PURPOSE**

Casts given DOMComment to a DOMNode

**SYNTAX**

`FUNCTION makeNode(com DOMComment) RETURN DOMNode;`

**PARAMETERS**

`com` (IN)- DOMComment to cast

**RETURNS**

The DOMComment

## DOM Implementation Methods

**FUNCTION**

`isNull(di DOMImplementation)`

**RETURN**

BOOLEAN;

**PURPOSE**

Checks if the given DOM Implementation is null

**SYNTAX**

`FUNCTION isNull(di DOMImplementation) RETURN BOOLEAN;`

**PARAMETERS**

`di` (IN)- DOMImplementation to check



**RETURNS**

Whether given DOM Implementation is null: TRUE - is null, FALSE - is not null

**Document Fragment Methods****FUNCTION**

```
isNull(df DOMDocumentFragment)
```

**RETURN**

BOOLEAN;

**PURPOSE**

Checks if the given DOM DocumentFragment is null

**SYNTAX**

```
FUNCTION isNull(df DOMDocumentFragment) RETURN BOOLEAN;
```

**PARAMETERS**

df (IN)- DOMDocumentFragment to check

**RETURNS**

Whether given DOM DocumentFragment is null: TRUE - is null, FALSE - is not null

**FUNCTION**

```
makeNode(df DOMDocumentFragment)
```

**RETURN**

DOMComment;

**PURPOSE**

Casts given DOMDocumentFragment to a DOMNode

**SYNTAX**

```
FUNCTION makeNode(df DOMDocumentFragment) RETURN DOMNode;
```

### PARAMETERS

df (IN)- DOMDocumentFragment to cast

### RETURNS

The DOMNode

## Document Type Methods

FUNCTION **isNull(dt DOMDocumentType)** RETURN BOOLEAN;

### PURPOSE

Checks if the given DOM DocumentType is null

### SYNTAX

```
FUNCTION isNull(dt DOMDocumentType) RETURN BOOLEAN;
```

### PARAMETERS

dt (IN)- DOMDocumentType to check

### RETURNS

Whether given DOM DocumentType is null: TRUE - is null, FALSE - is not null

---

FUNCTION **makeNode(dt DOMDocumentType)** RETURN DOMNode;

### PURPOSE

Casts given DOMDocumentType to a DOMNode

### SYNTAX

```
FUNCTION makeNode(dt DOMDocumentType) RETURN DOMNode;
```

### PARAMETERS

dt (IN)- DOMDocumentType to cast

### RETURNS

The DOMNode

**FUNCTION****findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN)****RETURN**

DOMEntity;

**PURPOSE**

Finds an entity in the given DTD

**SYNTAX**

```
FUNCTION findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN) RETURN  
DOMEntity;
```

**PARAMETERS**

```
dt      (IN)-  DTD  
name    (IN)-  entity to find  
par     (IN)-  TRUE - parameter entity, FALSE - normal entity
```

**RETURNS**

The DOMEntity, if found.

**FUNCTION****findNotation(dt DOMDocumentType, name VARCHAR2)****RETURN**

DOMNotation;

**PURPOSE**

Finds a notation in the given DTD

**SYNTAX**

```
FUNCTION findNotation(dt DOMDocumentType, name VARCHAR2) RETURN DOMNotation;
```

**PARAMETERS**

```
dt      (IN)-  DTD  
name    (IN)-  notation to find
```

**RETURNS**

The DOMNotation, if found.

**FUNCTION**

**getPublicId(dt DOMDocumentType)**

**RETURN**

VARCHAR2;

**PURPOSE**

Return the public id of the given DTD

**SYNTAX**

FUNCTION getPublicId(dt DOMDocumentType) RETURN VARCHAR2;

**PARAMETERS**

dt (IN)- DTD

**RETURNS**

The public id

**FUNCTION**

**getSystemId(dt DOMDocumentType)**

**RETURN**

VARCHAR2;

**PURPOSE**

Return the system id of the given DTD

**SYNTAX**

FUNCTION getSystemId(dt DOMDocumentType) RETURN VARCHAR2;

**PARAMETERS**

dt (IN)- DTD

**RETURNS**

The system id

**PROCEDURE**

```
writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);
```

**PURPOSE**

Writes DTD to specified file using the database character set

**SYNTAX**

```
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);
```

**PARAMETERS**

dt (IN)- DOMDocumentType  
fileName (IN)- File to write to

**RETURNS**

Nothing

**PROCEDURE**

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);
```

**PURPOSE**

Writes DTD to specified buffer using the database character set

**SYNTAX****PROCEDURE**

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);
```

**PARAMETERS**

dt (IN)- DOMDocumentType  
buffer (OUT)- buffer to write to

**RETURNS**

Nothing

**PROCEDURE**

```
writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);
```

## PURPOSE

Writes DTD to specified clob using the database character set

## SYNTAX

```
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);
```

## PARAMETERS

dt (IN)- DOMDocumentType  
cl (OUT)- CLOB to write to

## RETURNS

Nothing

## PROCEDURE

```
writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset  
VARCHAR2);
```

## PURPOSE

Writes DTD to specified file using the given character set

## SYNTAX

## PROCEDURE

```
writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2);
```

## PARAMETERS

dt (IN)- DOMDocumentType  
fileName (IN)- File to write to  
charset (IN)- Character set

## RETURNS

Nothing

## PROCEDURE

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2,  
charset VARCHAR2);
```

**PURPOSE**

Writes DTD to specified buffer using the given character set

**SYNTAX****PROCEDURE**

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2, charset  
VARCHAR2);
```

**PARAMETERS**

dt (IN)- DOMDocumentType  
buffer (OUT)- buffer to write to  
charset (IN)- Character set

**RETURNS**

Nothing

**PROCEDURE**

```
writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset  
VARCHAR2);
```

**PURPOSE**

Writes DTD to specified clob using the given character set

**SYNTAX**

```
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset  
VARCHAR2);
```

**PARAMETERS**

dt (IN)- DOMDocumentType  
cl (OUT)- CLOB to write to  
charset (IN)- Character set

**RETURNS**

Nothing

## Element Methods

### **FUNCTION**

**isNull(elem DOMElement)**

### **RETURN**

BOOLEAN;

### **PURPOSE**

Checks if the given DOM Element is null

### **SYNTAX**

```
FUNCTION isNull(elem DOMElement) RETURN BOOLEAN;
```

### **PARAMETERS**

elem (IN)- DOMElement to check

### **RETURNS**

Whether given DOM Element is null: TRUE - is null, FALSE - is not null

### **FUNCTION**

**makeNode(elem DOMElement)**

### **RETURN**

DOMNode;

### **PURPOSE**

Casts given DOMElement to a DOMNode

### **SYNTAX**

```
FUNCTION makeNode(elem DOMElement) RETURN DOMNode;
```

### **PARAMETERS**

elem (IN)- DOMElement to cast



**RETURNS**

The DOMNode

**FUNCTION**

`getQualifiedName(elem DOMElement)`

**RETURN**

VARCHAR2;

**PURPOSE**

Returns the qualified name of the DOMElem

**SYNTAX****FUNCTION**

`getQualifiedName(elem DOMElement)`

**RETURN**

VARCHAR2;

**PARAMETERS**

`elem` (IN)- DOMElement

**RETURNS**

The qualified name

**FUNCTION**

`getNamespace(elem DOMElement)`

**RETURN**

VARCHAR2;

**PURPOSE**

Returns the namespace of the DOMElem

## SYNTAX

```
FUNCTION getNamespace(elem DOMElement) RETURN VARCHAR2;
```

## PARAMETERS

elem (IN)- DOMElement

## RETURNS

The namespace

## FUNCTION

```
getLocalName(elem DOMElement)
```

## RETURN

VARCHAR2;

## PURPOSE

Returns the local name of the DOMElem

## SYNTAX

```
FUNCTION getLocalName(elem DOMElement) RETURN VARCHAR2;
```

## PARAMETERS

elem (IN)- DOMElement

## RETURNS

The local name

```
FUNCTION getExpandedName(elem DOMElement) RETURN VARCHAR2;
```

## PURPOSE

Returns the expanded name of the DOMElem

## SYNTAX

```
FUNCTION getExpandedName(elem DOMElement) RETURN VARCHAR2;
```

## PARAMETERS

elem (IN)- DOMElement

## RETURNS

The expanded name

```
FUNCTION getChildrenByTagName(elem DOMElement, name varchar2)  
RETURN DOMNodeList;
```

## PURPOSE

Returns the children of the DOMElem having the given tag name

## SYNTAX

```
FUNCTION getChildrenByTagName(elem DOMElement, name varchar2) RETURN  
DOMNodeList;
```

## PARAMETERS

elem           (IN)- DOMElement  
name           (IN)- tag name (\* matches any tag)

## RETURNS

Children with the given tag name

```
FUNCTION getElementsByTagName(elem DOMElement, name varchar2, ns  
VARCHAR2) RETURN DOMNodeList;
```

## PURPOSE

Returns the element children of the DOMElem having the given tag name and namespace

## SYNTAX

```
FUNCTION getElementsByTagName(elem DOMElement, name varchar2, ns VARCHAR2 )  
RETURN DOMNodeList;
```

## PARAMETERS

elem           (IN)- DOMElement  
name           (IN)- tag name (\* matches any tag)  
ns              (IN)- namespace

## RETURNS

Elements with the given tag name and namespace

**FUNCTION** `getChildrenByTagName(elem DOMElem, name varchar2, ns VARCHAR2)` RETURN DOMNodeList;

### **PURPOSE**

Returns the children of the DOMElem having the given tag name and namespace

### **SYNTAX**

`FUNCTION getChildrenByTagName(elem DOMElem, name varchar2, ns VARCHAR2 )  
RETURN DOMNodeList;`

### **PARAMETERS**

elem        (IN)- DOMElem  
name        (IN)- tag name (\* matches any tag)  
ns          (IN)- namespace

### **RETURNS**

Children with the given tag name and namespace

**FUNCTION** `resolveNamespacePrefix(elem DOMElem, prefix VARCHAR2)`  
RETURN VARCHAR2;

### **PURPOSE**

Resolves the given namespace prefix

### **SYNTAX**

`FUNCTION resolveNamespacePrefix(elem DOMElem, prefix VARCHAR2) RETURN  
VARCHAR2;`

### **PARAMETERS**

elem        (IN)- DOMElem  
prefix      (IN)- namespace prefix

### **RETURNS**

The resolved namespace

## **Entity Methods**

**FUNCTION** `isNull(ent DOMEntity)` RETURN BOOLEAN;

**PURPOSE**

Checks if the given DOM Entity is null

**SYNTAX**

```
FUNCTION isNull(ent DOMEntity) RETURN BOOLEAN;
```

**PARAMETERS**

ent (IN)- DOMEntity to check

**RETURNS**

Whether given DOM Entity is null: TRUE - is null, FALSE - is not null

```
FUNCTION makeNode(ent DOMEntity) RETURN DOMNode;
```

**PURPOSE**

Casts given DOMEntity to a DOMNode

**SYNTAX**

```
FUNCTION makeNode(ent DOMEntity) RETURN DOMNode;
```

**PARAMETERS**

ent (IN)- DOMEntity to cast

**RETURNS**

The DOMNode

## Entity Reference Methods

```
FUNCTION isNull(eref DOMEntityReference) RETURN BOOLEAN;
```

**PURPOSE**

Checks if the given DOM EntityReference is null

**SYNTAX**

```
FUNCTION isNull(eref DOMEntityReference) RETURN BOOLEAN;
```

### PARAMETERS

eref (IN)- DOMEntityReference to check

### RETURNS

Whether given DOM EntityReference is null : TRUE - is null, FALSE - is not null

FUNCTION **makeNode(eref DOMEntityReference)** RETURN DOMNode;

### PURPOSE

Casts given DOMEntityReference to a DOMNode

### SYNTAX

```
FUNCTION makeNode(eref DOMEntityReference) RETURN DOMNode;
```

### PARAMETERS

eref (IN)- DOMEntityReference to cast

### RETURNS

The DOMNode

## Notation Methods

```
FUNCTION isNull(n DOMNotation) RETURN BOOLEAN;
```

### PURPOSE

Checks if the given DOM Notation is null

### SYNTAX

```
FUNCTION isNull(n DOMNotation) RETURN BOOLEAN;
```

### PARAMETERS

n (IN)- DOMNotation to check

### RETURNS

Whether given DOM Notation is null : TRUE - is null, FALSE - is not null

```
FUNCTION makeNode(n DOMNotation) RETURN DOMNode;
```

**PURPOSE**

Casts given DOMNotation to a DOMNode

**SYNTAX**

```
FUNCTION makeNode(n DOMNotation) RETURN DOMNode;
```

**PARAMETERS**

n (IN)- DOMNotation to cast

**RETURNS**

The DOMNode

## Processing Instruction Methods

```
FUNCTION isNull(pi DOMProcessingInstruction) RETURN BOOLEAN;
```

**PURPOSE**

Checks if the given DOM ProcessingInstruction is null

**SYNTAX**

```
FUNCTION isNull(pi DOMProcessingInstruction) RETURN BOOLEAN;
```

**PARAMETERS**

pi (IN)- DOMProcessingInstruction to check

**RETURNS**

Whether given DOM ProcessingInstruction is null : TRUE - is null, FALSE - is not null

```
FUNCTION makeNode(pi DOMProcessingInstruction) RETURN DOMNode;
```

**PURPOSE**

Casts given DOMProcessingInstruction to a DOMNode

### **SYNTAX**

```
FUNCTION makeNode(pi DOMProcessingInstruction) RETURN DOMNode;
```

### **PARAMETERS**

pi (IN)- DOMProcessingInstruction to cast

### **RETURNS**

The DOMNode

## **Text Methods**

```
FUNCTION isNull(t DOMText) RETURN BOOLEAN;
```

### **PURPOSE**

Checks if the given DOMText is null

### **SYNTAX**

```
FUNCTION isNull(t DOMText) RETURN BOOLEAN;
```

### **PARAMETERS**

t (IN)- DOMText to check

### **RETURNS**

Whether given DOMText is null : TRUE - is null, FALSE - is not null

```
FUNCTION makeNode(t DOMText) RETURN DOMNode;
```

### **PURPOSE**

Casts given DOMText to a DOMNode

### **SYNTAX**

```
FUNCTION makeNode(t DOMText) RETURN DOMNode;
```

### **PARAMETERS**

t (IN)- DOMText to cast



**RETURNS**

The DOMNode

**Document Methods**

FUNCTION **isNull(doc DOMDocument)** RETURN BOOLEAN;

**PURPOSE**

Checks if the given DOMDocument is null

**SYNTAX**

```
FUNCTION isNull(doc DOMDocument) RETURN BOOLEAN;
```

**PARAMETERS**

doc (IN)- DOMDocument to check

**RETURNS**

Whether given DOMDocument is null : TRUE - is null, FALSE - is not null

FUNCTION **makeNode(doc DOMDocument)** RETURN DOMNode;

**PURPOSE**

Cast given DOMDocument to a DOMNode

**SYNTAX**

```
FUNCTION makeNode(doc DOMDocument) RETURN DOMNode;
```

**PARAMETERS**

doc (IN)- DOMDocument to cast

**RETURNS**

The DOMNode

FUNCTION **newDOMDocument** RETURN DOMDocument;

## **PURPOSE**

Returns a new DOM Document instance

## **SYNTAX**

```
FUNCTION newDOMDocument RETURN DOMDocument;
```

## **PARAMETERS**

None

## **RETURNS**

A new DOMDocument instance

## **PROCEDURE freeDocument(doc DOMDocument)**

## **PURPOSE**

**Frees Document object**

## **SYNTAX**

```
PROCEDURE freeDocument(doc DOMDocument)
```

## **PARAMETERS**

doc (IN) - DOMDocument

```
FUNCTION getVersion(doc DOMDocument) RETURN VARCHAR2;
```

## **PURPOSE**

Gets version information for the XML document

## **SYNTAX**

```
FUNCTION getVersion(doc DOMDocument) RETURN VARCHAR2;
```

## **PARAMETERS**

doc (IN) - DOMDocument

## **RETURNS**

Version information

**PROCEDURE setVersion(doc DOMDocument, version VARCHAR2);**

### **PURPOSE**

Sets version information for the XML document

### **SYNTAX**

```
PROCEDURE setVersion(doc DOMDocument, version VARCHAR2);
```

### **PARAMETERS**

```
doc          (IN)-  DOMDocument
version      (IN)-  version information
```

### **RETURNS**

Nothing

**FUNCTION getCharset(doc DOMDocument) RETURN VARCHAR2;**

### **PURPOSE**

Gets character set of the XML document

### **SYNTAX**

```
FUNCTION getCharset(doc DOMDocument) RETURN VARCHAR2;
```

### **PARAMETERS**

```
doc          (IN)-  DOMDocument
```

### **RETURNS**

Character set of the XML document

**PROCEDURE setCharset(doc DOMDocument, charset VARCHAR2);**

### **PURPOSE**

Sets character set of the XML document

## SYNTAX

```
PROCEDURE setCharset(doc DOMDocument, charset VARCHAR2);
```

## PARAMETERS

```
doc      (IN)- DOMDocument  
charset  (IN)- Character set
```

## RETURNS

Nothing

```
FUNCTION getStandalone(doc DOMDocument) RETURN VARCHAR2;
```

## PURPOSE

Gets standalone information for the XML document

## SYNTAX

```
FUNCTION getStandalone(doc DOMDocument) RETURN VARCHAR2;
```

## PARAMETERS

```
doc      (IN)- DOMDocument
```

## RETURNS

Standalone information

```
PROCEDURE setStandalone(doc DOMDocument, value VARCHAR2);
```

## PURPOSE

Sets standalone information for the XML document

## SYNTAX

```
PROCEDURE setStandalone(doc DOMDocument, value VARCHAR2);
```

## PARAMETERS

```
doc      (IN)- DOMDocument  
value    (IN)- standalone information
```

**RETURNS**

Nothing

**PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2);**

**PURPOSE**

Writes XML document to specified file using the database character set

**SYNTAX**

```
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2);
```

**PARAMETERS**

```
doc      (IN)- DOMDocument  
fileName (IN)- File to write to
```

**RETURNS**

Nothing

**PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);**

**PURPOSE**

Writes XML document to specified buffer using the database character set

**SYNTAX**

```
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);
```

**PARAMETERS**

```
doc      (IN)- DOMDocument  
buffer   (OUT)- buffer to write to
```

**RETURNS**

Nothing

**PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB);**

## PURPOSE

Writes XML document to specified clob using the database character set

## SYNTAX

```
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB);
```

## PARAMETERS

doc (IN)- DOMDocument  
cl (OUT)- CLOB to write to

## RETURNS

Nothing

```
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2, charset  
VARCHAR2);
```

## PURPOSE

Writes XML document to specified file using the given character set

## SYNTAX

```
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
fileName (IN)- File to write to  
charset (IN)- Character set

## RETURNS

Nothing

```
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2,  
charset VARCHAR2);
```

## PURPOSE

Writes XML document to specified buffer using the given character set

**SYNTAX**

```
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset  
VARCHAR2);
```

**PARAMETERS**

doc (IN)- DOMDocument  
buffer (OUT)- buffer to write to  
charset (IN)- Character set

**RETURNS**

Nothing

```
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB, charset  
VARCHAR2);
```

**PURPOSE**

Writes XML document to specified clob using the given character set

**SYNTAX**

```
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);
```

**PARAMETERS**

doc (IN)- DOMDocument  
cl (OUT)- CLOB to write to  
charset (IN)- Character set

**RETURNS**

Nothing

```
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName  
VARCHAR2);
```

**PURPOSE**

Writes an external DTD to specified file using the database character set

## SYNTAX

```
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
fileName (IN)- File to write to

## RETURNS

Nothing

```
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT  
VARCHAR2);
```

## PURPOSE

Writes an external DTD to specified buffer using the database character set

## SYNTAX

```
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
buffer (OUT)- buffer to write to

## RETURNS

Nothing

```
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);
```

## PURPOSE

Writes an external DTD to specified clob using the database character set

## SYNTAX

```
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);
```

## PARAMETERS

doc (IN)- DOMDocument



cl (OUT)- CLOB to write to

## RETURNS

Nothing

**PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);**

## PURPOSE

Writes an external DTD to specified file using the given character set

## SYNTAX

```
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
fileName (IN)- File to write to  
charset (IN)- Character set

## RETURNS

Nothing

**PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);**

## PURPOSE

Writes an external DTD to specified buffer using the given character set

## SYNTAX

```
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
buffer (OUT)- buffer to write to  
charset (IN)- Character set

## RETURNS

Nothing

PROCEDURE **writeExternalDTDToClob**(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);

## PURPOSE

Writes an external DTD to specified clob using the given character set

## SYNTAX

```
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset
VARCHAR2);
```

## PARAMETERS

doc (IN)- DOMDocument  
cl (OUT)- CLOB to write to  
charset (IN)- Character set

## RETURNS

Nothing

## Interface org.w3c.dom.Attr

Public interface `Attr` extends `Node`.

The `Attr` interface represents an attribute in an `Element` object. Typically the allowable values for the attribute are defined in a document type definition.

`Attr` objects inherit the `Node` interface, but since they are not actually child nodes of the element they describe, the DOM does not consider them part of the document tree. Thus, the `Node` attributes `parentNode`, `previousSibling`, and `nextSibling` have a null value for `Attr` objects. The DOM takes the view that attributes are properties of elements rather than having a separate identity from the elements they are associated with; this should make it more efficient to implement such features as default attributes associated with all elements of a given type. Furthermore, `Attr` nodes may not be immediate children of a `DocumentFragment`. However, they can be associated with `Element` nodes contained within a `DocumentFragment`. In short, users of DOM need to be aware that `Attr` nodes have some things in common with other objects inheriting the

Node interface, but they also are quite distinct. The attribute's effective value is determined as follows: if this attribute has been explicitly assigned any value, that value is the attribute's effective value; otherwise, if there is a declaration for this attribute, and that declaration includes a default value, then that default value is the attribute's effective value; otherwise, the attribute does not exist on this element in the structure model until it has been explicitly added. Note that the `nodeValue` attribute on the `Attr` instance can also be used to retrieve the string version of the attribute's value(s). In XML, where the value of an attribute can contain entity references, the child nodes of the `Attr` node provide a representation in which entity references are not expanded. These child nodes may be either `Text` or `EntityReference` nodes. Because the attribute type may be unknown, there are no tokenized attribute values.

### **getName**

Returns the name of this attribute.

### **getSpecified**

If this attribute was explicitly given a value in the original document, this is `true`; otherwise, it is `false`.

### **getValue**

On retrieval, the value of the attribute is returned as a string.

### **setValue**

Enter an appropriate value.

## **Abstracts**

### **getName**

```
public abstract String getName()
```

Returns the name of this attribute.

### **getSpecified**

```
public abstract boolean getSpecified()
```

If this attribute was explicitly given a value in the original document, this is `true`; otherwise, it is `false`. Note that the implementation is in charge of this attribute, not the user. If the user changes the value of the attribute (even if it ends up having the same value as the default value) then the `specified` flag is automatically

flipped to `true`. To re-specify the attribute as the default value from the DTD, the user must delete the attribute. The implementation will then make a new attribute available with `specified` set to `false` and the default value (if one exists). In summary: If the attribute has an assigned value in the document then `specified` is `true`, and the value is the assigned value. If the attribute has no assigned value in the document and has a default value in the DTD, then `specified` is `false`, and the value is the default value in the DTD. If the attribute has no assigned value in the document and has a value of `#IMPLIED` in the DTD, then the attribute does not appear in the structure model of the document.

### getValue

```
public abstract String getValue()
```

On retrieval, the value of the attribute is returned as a string. Character and general entity references are replaced with their values.

On setting, this creates a `Text` node with the unparsed contents of the string.

### setValue

```
public abstract void setValue(String value)
```

## Interface org.w3c.dom.CDATASection

Public interface **CDATASection** extends `Text`.

CDATA sections are used to escape blocks of text containing characters that would otherwise be regarded as markup. The only delimiter that is recognized in a CDATA section is the `]]>` string that ends the CDATA section. CDATA sections can not be nested. The primary purpose is for including material such as XML fragments, without needing to escape all the delimiters.

The `DOMString` attribute of the `Text` node holds the text that is contained by the CDATA section. Note that this may contain characters that need to be escaped outside of CDATA sections and that, depending on the character encoding ("charset") chosen for serialization, it may be impossible to write out some characters as part of a CDATA section.

The `CDATASection` interface inherits the `CharacterData` interface through the `Text` interface. Adjacent `CDATASection` nodes are not merged by use of the `Element.normalize()` method.

## Interface org.w3c.dom.CharacterData

Public interface **CharacterData** extends `Node`.

The `CharacterData` interface extends `Node` with a set of attributes and methods for accessing character data in the DOM. For clarity this set is defined here rather than on each object that uses these attributes and methods. No DOM objects correspond directly to `CharacterData`, though `Text` and others do inherit the interface from it. All `offsets` in this interface start from 0.

### **appendData(String)**

Append the string to the end of the character data of the node.

### **DeleteData(int, int)**

Remove a range of characters from the node.

### **getData()**

The character data of the node that implements this interface.

### **getLength()**

The number of characters that are available through `data` and the `substringData` method below.

### **insertData(int, String)**

Insert a string at the specified character offset.

### **replaceData(int, int, String)**

Replace the characters starting at the specified character offset with the specified string.

### **setData(String)**

### **substringData(int, int)**

Extracts a range of data from the node.

## getData

```
public abstract String getData() throws DOMException
```

The character data of the node that implements this interface. The DOM implementation may not put arbitrary limits on the amount of data that may be stored in a `CharacterData` node. However, implementation limits may mean that the entirety of a node's data may not fit into a single `DOMString`. In such cases, the user may call `substringData` to retrieve the data in appropriately sized pieces.

### THROWS

`DOMException`

`NO_MODIFICATION_ALLOWED_ERR`: Raised when the node is read only.

### THROWS

`DOMException`

`DOMSTRING_SIZE_ERR`: Raised when it would return more characters than fit in a `DOMString` variable on the implementation platform.

## Abstracts

### setData

```
public abstract void setData(String data) throws DOMException
```

### getLength

```
public abstract int getLength()
```

The number of characters that are available through `data` and the `substringData` method below. This may have the value zero, i.e., `CharacterData` nodes may be empty.

### substringData

```
public abstract String substringData(int offset,  
                                     int count) throws DOMException
```

Extracts a range of data from the node.

### PARAMETERS

`offset` - Start offset of substring to extract.

`count` - The number of characters to extract.

**RETURNS**

The specified substring. If the sum of `offset` and `count` exceeds the `length`, then all characters to the end of the data are returned.

**THROWS**

`DOMException`

`INDEX_SIZE_ERR`: Raised if the specified offset is negative or greater than the number of characters in `data`, or if the specified `count` is negative.

`DOMSTRING_SIZE_ERR`: Raised if the specified range of text does not fit into a `DOMString`.

**appendData**

```
public abstract void appendData(String arg) throws DOMException
```

Append the string to the end of the character data of the node. Upon success, `data` provides access to the concatenation of `data` and the `DOMString` specified.

**PARAMETERS**

`arg` - The `DOMString` to append.

**THROWS**

`DOMException`

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

**insertData**

```
public abstract void insertData(int offset,  
                               String arg) throws DOMException
```

Insert a string at the specified character offset.

**PARAMETERS**

`offset` - The character offset at which to insert.

`arg` - The `DOMString` to insert.

**THROWS**

`DOMException`

**INDEX\_SIZE\_ERR**: Raised if the specified offset is negative or greater than the number of characters in data.

**NO\_MODIFICATION\_ALLOWED\_ERR**: Raised if this node is read only.

## deleteData

```
public abstract void deleteData(int offset,  
                               int count) throws DOMException
```

Remove a range of characters from the node. Upon success, data and length reflect the change.

### PARAMETERS

**offset** - The offset from which to remove characters.

**count** - The number of characters to delete. If the sum of `offset` and `count` exceeds `length` then all characters from `offset` to the end of the data are deleted.

### THROWS

DOMException

**INDEX\_SIZE\_ERR**: Raised if the specified offset is negative or greater than the number of characters in data, or if the specified `count` is negative.

**NO\_MODIFICATION\_ALLOWED\_ERR**: Raised if this node is read only.

## replaceData

```
public abstract void replaceData(int offset,  
                                int count,  
                                String arg) throws DOMException
```

Replace the characters starting at the specified character offset with the specified string.

### PARAMETERS

**offset** - The offset from which to start replacing.

**count** - The number of characters to replace. If the sum of `offset` and `count` exceeds `length`, then all characters to the end of the data are replaced (i.e., the effect is the same as a `remove` method call with the same range, followed by an `append` method invocation).

**arg** - The DOMString with which the range must be replaced.



## THROWS

`DOMException`

`INDEX_SIZE_ERR`: Raised if the specified offset is negative or greater than the number of characters in `data`, or if the specified `count` is negative.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

## Interface `org.w3c.dom.Comment`

Public interface **Comment** extends `Node`.

This represents the content of a comment, i.e., all the characters between the starting '`<!--`' and ending '`-->`'. Note that this is the definition of a comment in XML, and, in practice, HTML, although some HTML tools may implement the full SGML comment structure.

## Interface `org.w3c.dom.Document`

Public interface **Document** extends `Node`.

The `Document` interface represents the entire HTML or XML document. Conceptually, it is the root of the document tree, and provides the primary access to the document's data.

Since elements, text nodes, comments, processing instructions, etc. cannot exist outside the context of a `Document`, the `Document` interface also contains the factory methods needed to create these objects. The `Node` objects created have a `ownerDocument` attribute which associates them with the `Document` within whose context they were created.

### **`createAttribute(String)`**

Creates an `Attr` of the given name.

### **`createCDATASection(String)`**

Creates a `CDATASection` node whose value is the specified string.

### **`createComment(String)`**

Creates a `Comment` node given the specified string.

### **createDocumentFragment()**

Creates an empty `DocumentFragment` object.

### **createElement(String)**

Creates an element of the type specified.

### **createEntityReference(String)**

Creates an `EntityReference` object.

### **createProcessingInstruction(String, String)**

Creates a `ProcessingInstruction` node given the specified name and data strings.

### **createTextNode(String)**

Creates a `Text` node given the specified string.

### **getDocType()**

The Document Type Declaration (see `DocumentType`) associated with this document.

### **getDocumentElement()**

This is a convenience attribute that allows direct access to the child node that is the root element of the document.

### **getElementsByTagName(String)**

Returns a `NodeList` of all the `Elements` with a given tag name in the order in which they would be encountered in a preorder traversal of the `Document` tree.

### **getImplementation()**

The `DOMImplementation` object that handles this document.

## **Abstracts**

### **getDoctype**

```
public abstract getDoctype()
```

The Document Type Declaration (see `DocumentType`) associated with this document. For HTML documents as well as XML documents without a document type declaration this returns `null`. The DOM Level 1 does not support editing the Document Type Declaration, therefore `docType` cannot be altered in any way.

## getImplementation

```
public abstract getImplementation()
```

The `DOMImplementation` object that handles this document. A DOM application may use objects from multiple implementations.

## getDocumentElement

```
public abstract getDocumentElement()
```

This is a convenience attribute that allows direct access to the child node that is the root element of the document. For HTML documents, this is the element with the `tagName` "HTML".

## createElement

```
public abstract createElement(String tagName) throws DOMException
```

Creates an element of the type specified. Note that the instance returned implements the `Element` interface, so attributes can be specified directly on the returned object.

### PARAMETERS

`tagName` - The name of the element type to instantiate. For XML, this is case-sensitive. For HTML, the `tagName` parameter may be provided in any case, but it must be mapped to the canonical uppercase form by the DOM implementation.

### RETURNS

A new `Element` object.

### THROWS

`DOMException`

`INVALID_CHARACTER_ERR`: Raised if the specified name contains an invalid character.

## Abstracts

### createDocumentFragment

```
public abstract createDocumentFragment()  
Creates an empty DocumentFragment object.
```

#### RETURNS

A new DocumentFragment.

### createTextNode

```
public abstract createTextNode(String data)  
Creates a Text node given the specified string.
```

#### PARAMETERS

data - The data for the node.

#### RETURNS

The new Text object.

### createComment

```
public abstract createComment(String data)  
Creates a Comment node given the specified string.
```

#### PARAMETERS

data - The data for the node.

#### RETURNS

The new Comment object.

### createCDATASection

```
public abstract createCDATASection(String data) throws DOMException  
Creates a CDATASection node whose value is the specified string.
```

#### PARAMETERS

data - The data for the CDATASection contents.

**RETURNS**

The new `CDATASection` object.

**THROWS**

`DOMException`

`NOT_SUPPORTED_ERR`: Raised if this document is an HTML document.

**createProcessingInstruction**

```
public abstract createProcessingInstruction(String target,String data) throws  
DOMException
```

Creates a `ProcessingInstruction` node given the specified name and data strings.

**PARAMETERS**

target - The target part of the processing instruction.

data - The data for the node.

**RETURNS**

The new `ProcessingInstruction` object.

**THROWS**

`DOMException`

`INVALID_CHARACTER_ERR`: Raised if an invalid character is specified.

`NOT_SUPPORTED_ERR`: Raised if this document is an HTML document.

**createAttribute**

```
public abstract createAttribute(String name) throws DOMException
```

Creates an `Attr` of the given name. Note that the `Attr` instance can then be set on an `Element` using the `setAttribute` method.

**PARAMETERS**

name - The name of the attribute.

**RETURNS**

A new `Attr` object.

### THROWS

DOMException

INVALID\_CHARACTER\_ERR: Raised if the specified name contains an invalid character.

### createEntityReference

`public abstract createEntityReference(String name) throws DOMException`  
Creates an EntityReference object.

### PARAMETERS

name - The name of the entity to reference.

### RETURNS

The new EntityReference object.

### THROWS

INVALID\_CHARACTER\_ERR: Raised if the specified name contains an invalid character.

NOT\_SUPPORTED\_ERR: Raised if this document is an HTML document.

### getElementsByTagName

`public abstract getElementsByTagName(String tagname)`  
Returns a NodeList of all the Elements with a given tag name in the order in which they would be encountered in a preorder traversal of the Document tree.

### PARAMETERS

tagname - The name of the tag to match on. The special value "\*" matches all tags.

### RETURNS

A new NodeList object containing all the matched Elements.

## Interface org.w3c.dom.DocumentFragment

Public interface **DocumentFragment** extends Node.

`DocumentFragment` is a "lightweight" or "minimal" `Document` object. It is very common to want to be able to extract a portion of a document's tree or to create a new fragment of a document. Imagine implementing a user command like cut or rearranging a document by moving fragments around. It is desirable to have an object which can hold such fragments and it is quite natural to use a `Node` for this purpose. While it is true that a `Document` object could fulfil this role, a `Document` object can potentially be a heavyweight object, depending on the underlying implementation. What is really needed for this is a very lightweight object. `DocumentFragment` is such an object.

Furthermore, various operations -- such as inserting nodes as children of another `Node` -- may take `DocumentFragment` objects as arguments; this results in all the child nodes of the `DocumentFragment` being moved to the child list of this node.

The children of a `DocumentFragment` node are zero or more nodes representing the tops of any sub-trees defining the structure of the document.

`DocumentFragment` nodes do not need to be well-formed XML documents (although they do need to follow the rules imposed upon well-formed XML parsed entities, which can have multiple top nodes). For example, a `DocumentFragment` might have only one child and that child node could be a `Text` node. Such a structure model represents neither an HTML document nor a well-formed XML document.

When a `DocumentFragment` is inserted into a `Document` (or indeed any other `Node` that may take children) the children of the `DocumentFragment` and not the `DocumentFragment` itself are inserted into the `Node`. This makes the `DocumentFragment` very useful when the user wishes to create nodes that are siblings; the `DocumentFragment` acts as the parent of these nodes so that the user can use the standard methods from the `Node` interface, such as `insertBefore()` and `appendChild()`.

## Interface org.w3c.dom.DocumentType

Public interface **DocumentType** extends `Node`.

Each `Document` has a `doctype` attribute whose value is either `null` or a `DocumentType` object. The `DocumentType` interface in the DOM Level 1 Core provides an interface to the list of entities that are defined for the document, and

little else because the effect of namespaces and the various XML scheme efforts on DTD representation are not clearly understood as of this writing.

The DOM Level 1 doesn't support editing `DocumentType` nodes.

### **getEntities()**

A `NamedNodeMap` containing the general entities, both external and internal, declared in the DTD.

### **getName()**

The name of DTD; i.e., the name immediately following the `DOCTYPE` keyword.

### **getNotations()**

A `NamedNodeMap` containing the notations declared in the DTD.

## **Abstracts**

### **getName**

```
public abstract String getName()
```

The name of DTD; i.e., the name immediately following the `DOCTYPE` keyword.

### **getEntities**

```
public abstract getEntities()
```

A `NamedNodeMap` containing the general entities, both external and internal, declared in the DTD. Duplicates are discarded. For example in:`<!DOCTYPE ex SYSTEM "ex.dtd" [ <!ENTITY foo "foo"> <!ENTITY bar "bar"> <!ENTITY % baz "baz">]> <ex/>` the interface provides access to `foo` and `bar` but not `baz`. Every node in this map also implements the `Entity` interface.

The DOM Level 1 does not support editing entities, therefore `entities` cannot be altered in any way.

### **getNotations**

```
public abstract getNotations()
```

A `NamedNodeMap` containing the notations declared in the DTD. Duplicates are discarded. Every node in this map also implements the `Notation` interface.

The DOM Level 1 does not support editing notations, therefore `notations` cannot be altered in any way.



## Class org.w3c.dom.DOMException

```
java.lang.Object
|
+--- java.lang.Throwable
      |
      +--- java.lang.Exception
            |
            +--- java.lang.RuntimeException
                  |
                  +--- org.w3c.dom.DOMException
```

Public abstract class **DOMException** extends RuntimeException.

DOM operations only raise exceptions in "exceptional" circumstances, i.e., when an operation is impossible to perform (either for logical reasons, because data is lost, or because the implementation has become unstable). In general, DOM methods return specific error values in ordinary processing situation, such as out-of-bound errors when using NodeList.

Implementations may raise other exceptions under other circumstances. For example, implementations may raise an implementation-dependent exception if a null argument is passed.

Some languages and object systems do not support the concept of exceptions. For such systems, error conditions may be indicated using native error reporting mechanisms. For some bindings, for example, methods may return error codes similar to those listed in the corresponding method descriptions.

### **INDEX\_SIZE\_ERR**

```
public static final short INDEX_SIZE_ERR
```

### **DOMSTRING\_SIZE\_ERR**

```
public static final short DOMSTRING_SIZE_ERR
```

### **HIERARCHY\_REQUEST\_ERR**

```
public static final short HIERARCHY_REQUEST_ERR
```

**WRONG\_DOCUMENT\_ERR**

public static final short WRONG\_DOCUMENT\_ERR

**INVALID\_CHARACTER\_ERR**

public static final short INVALID\_CHARACTER\_ERR

**NO\_DATA\_ALLOWED\_ERR**

public static final short NO\_DATA\_ALLOWED\_ERR

**NO\_MODIFICATION\_ALLOWED\_ERR**

public static final short NO\_MODIFICATION\_ALLOWED\_ERR

**NOT\_FOUND\_ERR**

public static final short NOT\_FOUND\_ERR

**NOT\_SUPPORTED\_ERR**

public static final short NOT\_SUPPORTED\_ERR

**INUSE\_ATTRIBUTE\_ERR**

public static final short INUSE\_ATTRIBUTE\_ERR

**DOMException**

```
public DOMException(short code,  
                    String message)
```

**Interface org.w3c.dom.DOMImplementation**

Public interface **DOMImplementation**

The `DOMImplementation` interface provides a number of methods for performing operations that are independent of any particular instance of the document object model.

The DOM Level 1 does not specify a way of creating a document instance, and hence document creation is an operation specific to an implementation. Future

Levels of the DOM specification are expected to provide methods for creating documents directly.

### hasFeature(String, String)

Tests if the DOM implementation implements a specific feature.

## Abstracts

### hasFeature

```
public abstract boolean hasFeature(String feature,
                                   String version)
```

Test if the DOM implementation implements a specific feature.

#### PARAMETERS

**feature** - The package name of the feature to test. In Level 1, the legal values are "HTML" and "XML" (case-insensitive).

**version** - This is the version number of the package name to test. In Level 1, this is the string "1.0". If the version is not specified, supporting any version of the feature will cause the method to return `true`.

#### RETURNS

`true` if the feature is implemented in the specified version, `false` otherwise.

## Interface org.w3c.dom.Element

Public interface **Element** extends `Node`.

By far the vast majority of objects (apart from text) that authors encounter when traversing a document are `Element` nodes. Assume the following XML document: `<elementExample id="demo"> <subelement1/> <subelement2><subsubelement/></subelement2> </elementExample>`

When represented using DOM, the top node is an `Element` node for "elementExample", which contains two child `Element` nodes, one for "subelement1" and one for "subelement2". "subelement1" contains no child nodes.

Elements may have attributes associated with them; since the `Element` interface inherits from `Node`, the generic `Node` interface method `getAttributes` may be used to retrieve the set of all attributes for an element. There are methods on the

`Element` interface to retrieve either an `Attr` object by name or an attribute value by name. In XML, where an attribute value may contain entity references, an `Attr` object should be retrieved to examine the possibly fairly complex sub-tree representing the attribute value. On the other hand, in HTML, where all attributes have simple string values, methods to directly access an attribute value can safely be used as a convenience.

### **getAttribute(String)**

Retrieves an attribute value by name.

### **getAttributeNode(String)**

Retrieves an `Attr` node by name.

### **getElementsByTagName(String)**

Returns a `NodeList` of all descendant elements with a given tag name, in the order in which they would be encountered in a preorder traversal of the `Element` tree.

### **getTagName()**

The name of the element.

### **normalize()**

Puts all `Text` nodes in the full depth of the sub-tree underneath this `Element` into a "normal" form where only markup (e.g., tags, comments, processing instructions, CDATA sections, and entity references) separates `Text` nodes, i.e., there are no adjacent `Text` nodes.

### **removeAttribute(String)**

Removes an attribute by name.

### **removeAttributeNode(Attr)**

Removes the specified attribute.

### **setAttribute(String, String)**

Adds a new attribute.

## setAttributeNode(Attr)

Adds a new attribute.

## Abstracts

### getTagName

```
public abstract String getTagName()
```

The name of the element. For example, in: `<elementExample id="demo"> ... </elementExample>`, `tagName` has the value `"elementExample"`. Note that this is case-preserving in XML, as are all of the operations of the DOM. The HTML DOM returns the `tagName` of an HTML element in the canonical uppercase form, regardless of the case in the source HTML document.

### getAttribute

```
public abstract String getAttribute(String name)
```

Retrieves an attribute value by name.

#### PARAMETERS

`name` - The name of the attribute to retrieve.

#### RETURNS

The `Attr` value as a string, or the empty string if that attribute does not have a specified or default value.

### setAttribute

```
public abstract void setAttribute(String name,  
                                 String value) throws DOMException
```

Adds a new attribute. If an attribute with that name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string, it is not parsed as it is being set. So any markup (such as syntax to be recognized as an entity reference) is treated as literal text, and needs to be appropriately escaped by the implementation when it is written out. In order to assign an attribute value that contains entity references, the user must create an `Attr` node plus any `Text` and `EntityReference` nodes, build the appropriate subtree, and use `setAttributeNode` to assign it as the value of an attribute.

### **PARAMETERS**

name - The name of the attribute to create or alter.

value - Value to set in string form.

### **THROWS**

`DOMException`

`INVALID_CHARACTER_ERR`: Raised if the specified name contains an invalid character.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

### **removeAttribute**

```
public abstract void removeAttribute(String name) throws DOMException
```

Removes an attribute by name. If the removed attribute has a default value it is immediately replaced.

### **PARAMETERS**

name - The name of the attribute to remove.

### **THROWS**

`DOMException`

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

### **getAttributeNode**

```
public abstract Attr getAttributeNode(String name)
```

Retrieves an `Attr` node by name.

### **PARAMETERS**

name - The name of the attribute to retrieve.

### **RETURNS**

The `Attr` node with the specified attribute name or `null` if there is no such attribute.

## Abstracts

### setAttributeNode

```
public abstract Attr setAttributeNode(Attr newAttr) throws
```

Adds a new attribute. If an attribute with that name is already present in the element, it is replaced by the new one.

#### PARAMETERS

`newAttr` - The `Attr` node to add to the attribute list.

#### RETURNS

If the `newAttr` attribute replaces an existing attribute with the same name, the previously existing `Attr` node is returned, otherwise `null` is returned.

#### THROWS

`DOMException`

`WRONG_DOCUMENT_ERR`: Raised if `newAttr` was created from a different document than the one that created the element.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

`INUSE_ATTRIBUTE_ERR`: Raised if `newAttr` is already an attribute of another `Element` object. The DOM user must explicitly clone `Attr` nodes to re-use them in other elements.

### removeAttributeNode

```
public abstract Attr removeAttributeNode(Attr oldAttr) throws DOMException
```

Removes the specified attribute.

#### PARAMETERS

`oldAttr` - The `Attr` node to remove from the attribute list. If the removed `Attr` has a default value it is immediately replaced.

#### RETURNS

The `Attr` node that was removed.

#### THROWS

`DOMException`

**NO\_MODIFICATION\_ALLOWED\_ERR**: Raised if this node is read only.  
**NOT\_FOUND\_ERR**: Raised if `oldAttr` is not an attribute of the element.

## getElementsByTagName

```
public abstract NodeList getElementsByTagName(String name)
```

Returns a `Nodelist` of all descendant elements with a given tag name, in the order in which they would be encountered in a preorder traversal of the `Element` tree.

### PARAMETERS

`name` - The name of the tag to match on. The special value "\*" matches all tags.

### RETURNS

A list of matching `Element` nodes.

## normalize

```
public abstract void normalize()
```

Puts all `Text` nodes in the full depth of the sub-tree underneath this `Element` into a "normal" form where only markup (e.g., tags, comments, processing instructions, CDATA sections, and entity references) separates `Text` nodes, i.e., there are no adjacent `Text` nodes. This can be used to ensure that the DOM view of a document is the same as if it were saved and re-loaded, and is useful when operations (such as `XPointer` lookups) that depend on a particular document tree structure are to be used.

## Interface org.w3c.dom.Entity

Public interface **Entity** extends `Node`.

This interface represents an entity, either parsed or unparsed, in an XML document. Note that this models the entity itself not the entity declaration. `Entity` declaration modeling has been left for a later Level of the DOM specification.

The `nodeName` attribute that is inherited from `Node` contains the name of the entity.

An XML processor may choose to completely expand entities before the structure model is passed to the DOM; in this case there will be no `EntityReference` nodes in the document tree.

XML does not mandate that a non-validating XML processor read and process entity declarations made in the external subset or declared in external parameter entities. This means that parsed entities declared in the external subset need not be



expanded by some classes of applications, and that the replacement value of the entity may not be available. When the replacement value is available, the corresponding `Entity` node's child list represents the structure of that replacement text. Otherwise, the child list is empty.

The resolution of the children of the `Entity` (the replacement value) may be lazily evaluated; actions by the user (such as calling the `childNodes` method on the `Entity Node`) are assumed to trigger the evaluation.

The DOM Level 1 does not support editing `Entity` nodes; if a user wants to make changes to the contents of an `Entity`, every related `EntityReference` node has to be replaced in the structure model by a clone of the `Entity`'s contents, and then the desired changes must be made to each of those clones instead. All the descendants of an `Entity` node are read only.

An `Entity` node does not have any parent.

### **getNotationName()**

For unparsed entities, the name of the notation for the entity.

### **getPublicId()**

The public identifier associated with the entity, if specified.

### **getSystemId()**

The system identifier associated with the entity, if specified.

## **Abstracts**

### **getPublicId**

```
public abstract String getPublicId()
```

The public identifier associated with the entity, if specified. If the public identifier was not specified, this is `null`.

### **getSystemId**

```
public abstract String getSystemId()
```

The system identifier associated with the entity, if specified. If the system identifier was not specified, this is `null`.

## getNotationName

```
public abstract String getNotationName()
```

For unparsed entities, the name of the notation for the entity. For parsed entities, this is `null`.

## Interface org.w3c.dom.EntityReference

Public interface **EntityReference** extends `Node`.

`EntityReference` objects may be inserted into the structure model when an entity reference is in the source document, or when the user wishes to insert an entity reference. Note that character references and references to predefined entities are considered to be expanded by the HTML or XML processor so that characters are represented by their Unicode equivalent rather than by an entity reference. Moreover, the XML processor may completely expand references to entities while building the structure model, instead of providing `EntityReference` objects. If it does provide such objects, then for a given `EntityReference` node, it may be that there is no `Entity` node representing the referenced entity; but if such an `Entity` exists, then the child list of the `EntityReference` node is the same as that of the `Entity` node. As with the `Entity` node, all descendants of the `EntityReference` are read only.

The resolution of the children of the `EntityReference` (the replacement value of the referenced `Entity`) may be lazily evaluated; actions by the user (such as calling the `childNodes` method on the `EntityReference` node) are assumed to trigger the evaluation.

## Interface org.w3c.dom.NamedNodeMap

Public interface **NamedNodeMap**

Objects implementing the `NamedNodeMap` interface are used to represent collections of nodes that can be accessed by name. Note that `NamedNodeMap` does not inherit from `NodeList`; `NamedNodeMaps` are not maintained in any particular order. Objects contained in an object implementing `NamedNodeMap` may also be accessed by an ordinal index, but this is simply to allow convenient enumeration of the contents of a `NamedNodeMap`, and does not imply that the DOM specifies an order to these `Nodes`.

## getLength()

The number of nodes in the map.

**getNamedItem(String)**

Retrieves a node specified by name.

**item(int)**

Returns the `index`th item in the map.

**removeNamedItem(String)**

Removes a node specified by name.

**setNamedItem(Node)**

Adds a node using its `nodeName` attribute.

**Abstracts****getNamedItem**

```
public abstract Node getNamedItem(String name)
```

Retrieves a node specified by name.

**PARAMETERS**

`name` - Name of a node to retrieve.

**RETURNS**

A `Node` (of any type) with the specified name, or `null` if the specified name did not identify any node in the map.

**setNamedItem**

```
public abstract Node setNamedItem(Node arg) throws DOMException
```

Adds a node using its `nodeName` attribute.

As the `nodeName` attribute is used to derive the name which the node must be stored under, multiple nodes of certain types (those that have a "special" string value) cannot be stored as the names would clash. This is seen as preferable to allowing nodes to be aliased.

**PARAMETERS**

`arg` - A node to store in a named node map. The node will later be accessible using the value of the `nodeName` attribute of the node. If a node with that name is already present in the map, it is replaced by the new one.

**RETURNS**

If the new `Node` replaces an existing node with the same name the previously existing `Node` is returned, otherwise `null` is returned.

**THROWS**

`DOMException`

`WRONG_DOCUMENT_ERR`: Raised if `arg` was created from a different document than the one that created the `NamedNodeMap`.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this `NamedNodeMap` is read only.

`INUSE_ATTRIBUTE_ERR`: Raised if `arg` is an `Attr` that is already an attribute of another `Element` object. The DOM user must explicitly clone `Attr` nodes to re-use them in other elements.

**removeNamedItem**

```
public abstract Node removeNamedItem(String name) throws DOMException
```

Removes a node specified by name. If the removed node is an `Attr` with a default value it is immediately replaced.

**PARAMETERS**

`name` - The name of a node to remove.

**RETURNS**

The node removed from the map or `null` if no node with such a name exists.

**THROWS**

`DOMException`

`NOT_FOUND_ERR`: Raised if there is no node named `name` in the map.

**item**

```
public abstract Node item(int index)
```

Returns the `index`th item in the map. If `index` is greater than or equal to the number of nodes in the map, this returns `null`.

**PARAMETERS**

index - Index into the map.

**RETURNS**

The node at the `index`th position in the `NamedNodeMap`, or `null` if that is not a valid index.

**getLength**

```
public abstract int getLength()
```

The number of nodes in the map. The range of valid child node indices is 0 to `length-1` inclusive.

**Interface org.w3c.dom.Node****Public interface Node**

The `Node` interface is the primary datatype for the entire Document Object Model. It represents a single node in the document tree. While all objects implementing the `Node` interface expose methods for dealing with children, not all objects implementing the `Node` interface may have children. For example, `Text` nodes may not have children, and adding children to such nodes results in a `DOMException` being raised.

The attributes `nodeName`, `nodeValue` and `attributes` are included as a mechanism to get at node information without casting down to the specific derived interface. In cases where there is no obvious mapping of these attributes for a specific `nodeType` (e.g., `nodeValue` for an `Element` or `attributes` for a `Comment`), this returns `null`. Note that the specialized interfaces may contain additional and more convenient mechanisms to get and set the relevant information.

**appendChild(Node)**

Adds the node `newChild` to the end of the list of children of this node.

**cloneNode(boolean)**

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.

### **getAttributes()**

A `NamedNodeMap` containing the attributes of this node (if it is an `Element`) or `null` otherwise.

### **getChildNodes()**

A `NodeList` that contains all children of this node.

### **getFirstChild()**

The first child of this node.

### **getLastChild()**

The last child of this node.

### **getNextSibling()**

The node immediately following this node.

### **getNodeName()**

The name of this node, depending on its type; see the table above.

### **getNodeType()**

A code representing the type of the underlying object, as defined above.

### **getNodeValue()**

The value of this node, depending on its type; see the table above.

### **getOwnerDocument()**

The `Document` object associated with this node.

### **getParentNode()**

The parent of this node.

### **getPreviousSibling()**

The node immediately preceding this node.

**hasChildNodes()**

This is a convenience method to allow easy determination of whether a node has any children.

**insertBefore(Node, Node)**

Inserts the node `newChild` before the existing child node `refChild`.

**removeChild(Node)**

Removes the child node indicated by `oldChild` from the list of children, and returns it.

**replaceChild(Node, Node)**

Replaces the child node `oldChild` with `newChild` in the list of children, and returns the `oldChild` node.

**setNodeValue(String)****ELEMENT\_NODE**

```
public static final short ELEMENT_NODE
```

**ATTRIBUTE\_NODE**

```
public static final short ATTRIBUTE_NODE
```

**TEXT\_NODE**

```
public static final short TEXT_NODE
```

**CDATA\_SECTION\_NODE**

```
public static final short CDATA_SECTION_NODE
```

**ENTITY\_REFERENCE\_NODE**

```
public static final short ENTITY_REFERENCE_NODE
```

**ENTITY\_NODE**

```
public static final short ENTITY_NODE
```

## PROCESSING\_INSTRUCTION\_NODE

```
public static final short PROCESSING_INSTRUCTION_NODE
```

## COMMENT\_NODE

```
public static final short COMMENT_NODE
```

## DOCUMENT\_NODE

```
public static final short DOCUMENT_NODE
```

## DOCUMENT\_TYPE\_NODE

```
public static final short DOCUMENT_TYPE_NODE
```

## DOCUMENT\_FRAGMENT\_NODE

```
public static final short DOCUMENT_FRAGMENT_NODE
```

## NOTATION\_NODE

```
public static final short NOTATION_NODE
```

## getNodeName

```
public abstract String getNodeName()
```

The name of this node, depending on its type; see the table above.

## getNodeValue

```
public abstract String getNodeValue() throws
```

The value of this node, depending on its type; see the table above.

## THROWS

DOMException

NO\_MODIFICATION\_ALLOWED\_ERR: Raised when the node is read only.

## THROWS

DOMSTRING\_SIZE\_ERR: Raised when it would return more characters than fit in a DOMString variable on the implementation platform.



**setNodeValue**

```
public abstract void setNodeValue(String nodeValue) throws DOMException
```

**getNodeTypes**

```
public abstract short getNodeType()
```

A code representing the type of the underlying object, as defined above.

**getParentNode**

```
public abstract Node getParentNode()
```

The parent of this node. All nodes, except Document, DocumentFragment, and Attr may have a parent. However, if a node has just been created and not yet added to the tree, or if it has been removed from the tree, this is null.

**getChildNodes**

```
public abstract NodeList getChildNodes()
```

A NodeList that contains all children of this node. If there are no children, this is a NodeList containing no nodes. The content of the returned NodeList is "live" in the sense that, for instance, changes to the children of the node object that it was created from are immediately reflected in the nodes returned by the NodeList accessors; it is not a static snapshot of the content of the node. This is true for every NodeList, including the ones returned by the getElementByTagName method.

**getFirstChild**

```
public abstract Node getFirstChild()
```

The first child of this node. If there is no such node, this returns null.

**getLastChild**

```
public abstract Node getLastChild()
```

The last child of this node. If there is no such node, this returns null.

**getPreviousSibling**

```
public abstract Node getPreviousSibling()
```

The node immediately preceding this node. If there is no such node, this returns null.

**getNextSibling**

```
public abstract Node getNextSibling()
```

The node immediately following this node. If there is no such node, this returns null.

### getAttributes

```
public abstract NamedNodeMap getAttributes()
```

A `NamedNodeMap` containing the attributes of this node (if it is an `Element`) or null otherwise.

### getOwnerDocument

```
public abstract Document getOwnerDocument()
```

The `Document` object associated with this node. This is also the `Document` object used to create new nodes. When this node is a `Document` this is null.

### insertBefore

```
public abstract Node insertBefore(Node newChild,  
                                  Node refChild) throws DOMException
```

Inserts the node `newChild` before the existing child node `refChild`. If `refChild` is null, insert `newChild` at the end of the list of children.

If `newChild` is a `DocumentFragment` object, all of its children are inserted, in the same order, before `refChild`. If the `newChild` is already in the tree, it is first removed.

### PARAMETERS

`newChild` - The node to insert.

`refChild` - The reference node, i.e., the node before which the new node must be inserted.

### RETURNS

The node being inserted.

### THROWS

`DOMException`

**HIERARCHY\_REQUEST\_ERR**: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to insert is one of this node's ancestors.

**WRONG\_DOCUMENT\_ERR**: Raised if `newChild` was created from a different document than the one that created this node.

**NO\_MODIFICATION\_ALLOWED\_ERR**: Raised if this node is read only.

**NOT\_FOUND\_ERR**: Raised if `refChild` is not a child of this node.

## replaceChild

```
public abstract Node replaceChild(Node newChild,  
                                  Node oldChild) throws DOMException
```

Replaces the child node `oldChild` with `newChild` in the list of children, and returns the `oldChild` node. If the `newChild` is already in the tree, it is first removed.

### PARAMETERS

`newChild` - The new node to put in the child list.

`oldChild` - The node being replaced in the list.

### RETURNS

The node replaced.

### THROWS

`DOMException`

**HIERARCHY\_REQUEST\_ERR**: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to put in is one of this node's ancestors.

**WRONG\_DOCUMENT\_ERR**: Raised if `newChild` was created from a different document than the one that created this node.

**NO\_MODIFICATION\_ALLOWED\_ERR**: Raised if this node is read only.

**NOT\_FOUND\_ERR**: Raised if `oldChild` is not a child of this node.

## removeChild

```
public abstract Node removeChild(Node oldChild) throws DOMException
```

Removes the child node indicated by `oldChild` from the list of children, and returns it.

### PARAMETERS

`oldChild` - The node being removed.

## RETURNS

The node removed.

## THROWS

`DOMException`

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

`NOT_FOUND_ERR`: Raised if `oldChild` is not a child of this node.

## appendChild

```
public abstract appendChild( newChild) throws DOMException
```

Adds the node `newChild` to the end of the list of children of this node. If the `newChild` is already in the tree, it is first removed.

## PARAMETERSPARAMETERS

`newChild` - The node to add.If it is a `DocumentFragment` object, the entire contents of the document fragment are moved into the child list of this node

## RETURNS

The node added.

## THROWS

`DOMException`

`HIERARCHY_REQUEST_ERR`: Raised if this node is of a type that does not allow children of the type of the `newChild` node, or if the node to append is one of this node's ancestors.

`WRONG_DOCUMENT_ERR`: Raised if `newChild` was created from a different document than the one that created this node.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.

## hasChildNodes

```
public abstract boolean hasChildNodes()
```

This is a convenience method to allow easy determination of whether a node has any children.

## RETURNS

`true` if the node has any children, `false` if the node has no children.

## cloneNode

```
public abstract Node cloneNode(boolean deep)
```

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (`parentNode` returns `null`). Cloning an `Element` copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child `Text` node. Cloning any other type of node simply returns a copy of this node.

### PARAMETERS

`deep` - If `true`, recursively clone the subtree under the specified node; if `false`, clone only the node itself (and its attributes, if it is an `Element`).

### RETURNS

The duplicate node.

## Interface org.w3c.dom.NodeList

Public interface **NodeList**

The `NodeList` interface provides the abstraction of an ordered collection of nodes, without defining or constraining how this collection is implemented.

The items in the `NodeList` are accessible via an integral index, starting from 0.

### getLenth()

The number of nodes in the list.

### item(int)

Returns the `index`th item in the collection.

## Abstracts

### item

```
public abstract Node item(int index)
```

Returns the `index`th item in the collection. If `index` is greater than or equal to the number of nodes in the list, this returns `null`.

### PARAMETERS

index - Index into the collection.

### RETURNS

The node at the `index`th position in the `NodeList`, or `null` if that is not a valid index.

### `getLength`

```
public abstract int getLength()
```

The number of nodes in the list. The range of valid child node indices is 0 to `length-1` inclusive.

## Interface org.w3c.dom.Notation

Public interface **Notation** extends `Node`.

This interface represents a notation declared in the DTD. A notation either declares, by name, the format of an unparsed entity (see section 4.7 of the XML 1.0 specification), or is used for formal declaration of Processing Instruction targets (see section 2.6 of the XML 1.0 specification). The `nodeName` attribute inherited from `Node` is set to the declared name of the notation.

The DOM Level 1 does not support editing `Notation` nodes; they are therefore read only.

A `Notation` node does not have any parent.

### `getPublicId()`

The public identifier of this notation.

### `getSystemId()`

The system identifier of this notation.

## Abstracts

### getPublicId

```
public abstract String getPublicId()
```

The public identifier of this notation. If the public identifier was not specified, this is `null`.

### getSystemId

```
public abstract String getSystemId()
```

The system identifier of this notation. If the system identifier was not specified, this is `null`.

## Interface org.w3c.dom.ProcessingInstruction

Public interface **ProcessingInstruction** extends `Node`.

The `ProcessingInstruction` interface represents a "processing instruction", used in XML as a way to keep processor-specific information in the text of the document.

### getData()

The content of this processing instruction.

### getTarget()

The target of this processing instruction.

### setData(String)

## Abstracts

### getTarget

```
public abstract String getTarget()
```

The target of this processing instruction. XML defines this as being the first token following the markup that begins the processing instruction.

## getData

```
public abstract String getData()
```

The content of this processing instruction. This is from the first non white space character after the target to the character immediately preceding the `?>`.

### THROWS

`DOMException`

`NO_MODIFICATION_ALLOWED_ERR`: Raised when the node is read only.

## setData

```
public abstract void setData(String data) throws
```

## Interface org.w3c.dom.Text

Public interface **Text** extends `CharacterData`.

The `Text` interface represents the textual content (termed character data in XML) of an `Element` or `Attr`. If there is no markup inside an element's content, the text is contained in a single object implementing the `Text` interface that is the only child of the element. If there is markup, it is parsed into a list of elements and `Text` nodes that form the list of children of the element.

When a document is first made available via the DOM, there is only one `Text` node for each block of text. Users may create adjacent `Text` nodes that represent the contents of a given element without any intervening markup, but should be aware that there is no way to represent the separations between these nodes in XML or HTML, so they will not (in general) persist between DOM editing sessions. The `normalize()` method on `Element` merges any such adjacent `Text` objects into a single node for each block of text; this is recommended before employing operations that depend on a particular document structure, such as navigation with `XPointers`.

## splitText(int)

Breaks this `Text` node into two `Text` nodes at the specified offset, keeping both in the tree as siblings.

## Abstracts

### splitText

```
public abstract Text splitText(int offset) throws DOMException
```



---

Breaks this `Text` node into two `Text` nodes at the specified `offset`, keeping both in the tree as siblings. This node then only contains all the content up to the `offset` point. And a new `Text` node, which is inserted as the next sibling of this node, contains all the content at and after the `offset` point.

### **PARAMETERS**

`offset` - The offset at which to split, starting from 0.

### **RETURNS**

The new `Text` node.

### **THROWS**

`DOMException`

`INDEX_SIZE_ERR`: Raised if the specified `offset` is negative or greater than the number of characters in `data`.

`NO_MODIFICATION_ALLOWED_ERR`: Raised if this node is read only.



# Part III

---

## XDK for C Packages

This section contains

- [Chapter 7, "XML Parser for C"](#)
- [Chapter 8, "XML Schema Processor for C"](#)



---

# XML Parser for C

This chapter describes the following sections:

- [Parser APIs](#)
- [XSLT API](#)
- [W3C SAX APIs](#)
- [W3C DOM APIs](#)
- [Namespace APIs](#)
- [Datatypes](#)

## Parser APIs

Extensible Markup Language (XML) describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

A software module called an XML processor is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the application.

This C implementation of the XML processor (or parser) followed the W3C XML specification (rev REC-xml-19980210) and included the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

The following is the default behavior of this parser:

- The character set encoding is UTF-8. If all your documents are ASCII, you are encouraged to set the encoding to US-ASCII for better performance.
- Messages are printed to stderr unless `msghdlr` is given.
- A parse tree which can be accessed by DOM APIs is built unless `saxcb` is set to use the SAX callback APIs. Note that any of the SAX callback functions can be set to NULL if not needed.
- The default behavior for the parser is to check that the input is well-formed but not to check whether it is valid. The flag `XML_FLAG_VALIDATE` can be set to validate the input. The default behavior for whitespace processing is to be fully conformant to the XML 1.0 spec, i.e. all whitespace is reported back to the application but it is indicated which whitespace is ignorable. However, some applications may prefer to set the `XML_FLAG_DISCARD_WHITESPACE` which will discard all whitespace between an end-element tag and the following start-element tag.

## Calling Sequence

Parsing a single document:

```
xmlinit, xmlparsexxx, xmlterm
```

Parsing multiple documents, but only the latest document needs to be available:

```
xmlinit, xmlparsexxx, xmlclean, xmlparsexxx, xmlclean ... xmlterm
```

Parsing multiple documents, all document data must be available:

```
xmlinit, xmlparsexxx, xmlparsexxx ... xmlterm
```

## Memory

The memory callback functions specified in `memcb` may be used if you wish to use your own memory allocation. If they are used, all of the functions should be specified.

The memory allocated for parameters passed to the SAX callbacks or for nodes and data stored with the DOM parse tree will not be freed until one of the following is done:

- `xmlparsexxx` is called to parse another document.
- `xmlclean` is called.
- `xmlterm` is called.

## Thread Safety

If threads are forked off somewhere in the midst of the `init-parse-terminate` sequence of calls, you will get unpredictable behavior and results.

## Function/Method Index

<code>xmlinit</code>	<code>XMLParser::xmlinit</code>	Initialize XML parser
<code>xmlclean</code>		Clean up memory used during parse
<code>xmlparse</code>		Parse a document specified by a URL

<code>xmlparsebuf</code>	Parse a document that's resident in memory
<code>xmlparsefile</code>	Parse a document from the filesystem
<code>xmlparsestream</code>	Parse a document from a user-defined stream
<code>xmlterm</code>	Shut down XML parser
<code>createDocument</code>	Create a new document
<code>isStandalone</code>	Return document's standalone flag
<code>isSingleChar</code>	Return single/multibyte encoding flag
<code>getEncoding</code>	Return name of document's encoding



---

## Functions

### xmlinit

#### Purpose

Initializes the XML parser. It must be called before any parsing can take place.

#### C Prototype

```
xmlctx *xmlinit(uword *err, const oratext *encoding,
               void (*msghdlr)(void *msgctx, const oratext *msg,
                               uword errcode),
               void *msgctx, const xmlsaxcb *saxcb, void *saxcbctx,
               const xmlmemcb *memcb, void *memcbctx, const oratext *lang);
```

#### Parameters

`err` (OUT) - The error, if any (C only)  
`encoding` (IN) - *default* character set encoding  
`msghdlr` (IN) - Error message handler function  
`msgctx` (IN) - Context for the error message handler  
`saxcb` (IN) - SAX callback structure filled with function pointers  
`saxcbctx` (IN) - Context for SAX callbacks  
`memcb` (IN) - Memory function callbacks  
`memcbctx` (IN) - Context for the memory function callbacks  
`lang` (IN) - Language for error messages

#### Comments

The C version of this call returns the XML context on success, and sets the user's `err` argument on error. As usual, a zero error code means success, non-zero indicates a problem.

This function should only be called once before starting the processing of one or more XML files. `xmlterm()` should be called after all processing of XML files has completed.

Error codes: `XMLERR_LEH_INIT`, `XMLERR_BAD_ENCODING`, `XMLERR-NLS_INIT`, `XMLERR_NO_MEMORY`, `XMLERR_NULL_PTR`

For C, all arguments may be NULL except for `err`. For C++, all arguments have default values and may be omitted if not needed.

By default, the character set encoding is UTF-8. If all your documents are ASCII, you are encouraged to set the encoding to US-ASCII for better performance.

By default, messages are printed to `stderr` unless `msghdlr` is given.

By default, a parse tree is built (accessible by DOM APIs) unless `saxcb` is set (in which case the SAX callback APIs are invoked). Note that any of the SAX callback functions can be set to NULL if not needed.

The memory callback functions `memcb` may be used if you wish to use your own memory allocation. If they are used, all of the functions should be specified.

The parameters `msgctx`, `saxcbctx`, and `memcbctx` are structures that you may define and use to pass information to your callback routines for the message handler, SAX functions, or memory functions, respectively. They should be set to NULL if your callback functions do not need any additional information passed in to them.

The `lang` parameter is not used currently and may be set to NULL. It will be used in future releases to determine the language of the error messages.

## xmlclean

### Purpose

Frees memory used during the previous parse.

### Syntax

```
void xmlclean(xmlctx *ctx);
```

### Parameters

`ctx` (IN) - The XML parser context

### Comments

This function is provided as a convenience for those who want to parse multiple documents using a single context. Before parsing the second and subsequent documents, call `xmlclean` to release memory used by the previous document.

Note that memory is reused internally after this call. Memory is not returned to the system until `xmlterminate`.

## xmlparse

### Purpose

Invokes the XML parser on an input document that is specified by a URL. The parser must have been initialized successfully with a call to `xmlinit` first.

### Syntax

```
uword xmlparse(xmlctx *ctx, const oratext *url,  
               const oratext *encoding, ub4 flags);
```

### Parameters

`ctx` (IN/OUT) - The XML parser context  
`url` (IN) - URL of XML document  
`encoding` (IN) - *default* character set encoding  
`flags` (IN) - what options to use

### Comments

Flag bits must be OR'd to override the default behavior of the parser. The following flag bits may be set:

- `XML_FLAG_VALIDATE` turns validation on. The default behavior is to not validate the input.
- `XML_FLAG_DISCARD_WHITESPACE` will discard whitespace where it appears to be insignificant. The default behavior for whitespace processing is to be fully conformant to the XML 1.0 spec, i.e. all whitespace is reported back to the application but it is indicated which whitespace is ignorable. However, some applications may prefer to set the `XML_FLAG_DISCARD_WHITESPACE` which will discard all whitespace between an end-element tag and the following start-element tag.
- `XML_FLAG_DTD_ONLY` tells the parser that the input is an external DTD only, not a complete document.
- `XML_FLAG_STOP_ON_WARNING` makes the parser stop immediately if any validation warnings occur. By default, validation warnings are printed but validation continues.

The memory passed to the SAX callbacks or stored with the DOM parse tree will not be freed until one of the following is done:

- `xmlparsexxx` is called to parse another document.

- `xmlclean` is called.
- `xmlterm` is called.

## `xmlparsebuf`

### **Purpose**

Invokes the XML parser on a document that is resident in memory. The parser must have been initialized successfully with a call to `xmlinit` first.

### **Syntax**

```
uword xmlparsebuf(xmlctx *ctx, const oratext *buffer, size_t len,  
                  const oratext *encoding, ub4 flags);
```

### **Parameters**

`ctx` (IN/OUT) - The XML parser context  
`buffer` (IN) - pointer to document in memory  
`len` (IN) - length of the buffer  
`encoding` (IN) - default character set encoding  
`flags` (IN) - what options to use

### **Comments**

This function is identical to `xmlparse` except that input is taken from the user's buffer instead of from a URI, file, etc.

## `xmlparsefile`

### **Purpose**

Invokes the XML parser on a document in the filesystem. The parser must have been initialized successfully with a call to `xmlinit` first.

### **Syntax**

```
uword xmlparsefile(xmlctx *ctx, const oratext *path,  
                  const oratext *encoding, ub4 flags);
```

### **Parameters**

`ctx` (IN/OUT) - The XML parser context  
`path` (IN) - filesystem path of document  
`encoding` (IN) - default character set encoding

flags (IN) - what options to use

### Comments

This function is identical to `xmlparse` except that input is taken from a file in the user's filesystem, instead of from a URL, memory buffer, etc.

## xmlparsestream

### Purpose

Invokes the XML parser on a document that is to be read from a user-defined stream. The parser must have been initialized successfully with a call to `xmlinit` first.

### Syntax

```
uword xmlparsestream(xmlctx *ctx, const void *stream,  
                    const oratext *encoding, ub4 flags);
```

### Parameters

ctx (IN/OUT) - The XML parser context  
stream (IN) - pointer to stream or stream context  
encoding (IN) - *default* character set encoding  
flags (IN) - what options to use

### Comments

This function is identical to `xmlparse` except that input is taken from a user-defined stream, instead of from a URL, file, etc. The I/O callback functions for access method `XMLACCESS_STREAM` must be set up first. The stream (or stream context) pointer will be available in each callback function as the `ptr_xmlihdl` memory of the `ihdl` structure. Its meaning and use are user-defined.

## xmlterm

### Purpose

Terminates the XML parser. It should be called after `xmlinit`, and before exiting the main program.

### Syntax

```
uword xmlterm(xmlctx *ctx);
```

### Parameters

ctx (IN) - the XML parser context

### Comments

This function will free all memory used by the parser and terminates the context, which may not then be reused (a new context must be created if additional parsing is to be done).

## createDocument

### Purpose

Creates a new document in memory.

### Syntax

```
xmlnode* createDocument(xmlctx *ctx)
```

### Parameters

ctx (IN) - the XML parser context

### Comments

This function is used when constructing a new document in memory. An XML document is always rooted in a node of type `DOCUMENT_NODE`. This function creates that root node and sets it in the context. There can be only one current document and hence only one document node; if one already exists, this function does nothing and returns `NULL`.

## isStandalone

### Purpose

Return value of document's *standalone* flag.

### Syntax

```
boolean isStandalone(xmlctx *ctx)
```

**Parameters**

ctx (IN) - the XML parser context

**Comments**

This function returns the boolean value of the document's standalone flag, as specified in the `<?xml?>` processing instruction.

**isSingleChar****Purpose**

Returns a flag which specifies whether the current document is encoded as single-byte characters (i.e. ASCII), or multi-byte characters (e.g. UTF-8).

**Syntax**

```
boolean isSingleChar(xmlctx *ctx)
```

**Parameters**

ctx (IN) - the XML parser context

**Comments**

Compare to `getEncoding`, which returns the actual name of the document's encoding.

**getEncoding****Purpose**

Returns the name of the current document's character encoding scheme (e.g., "ASCII", "UTF8", etc).

**Syntax**

```
oratext *getEncoding(xmlctx *ctx)
```

**Parameters**

ctx (IN) - the XML parser context

## Comments

Compare to `isSingleChar` which just returns a boolean flag saying whether the current encoding is single or multi-byte.

## XSLT API

XSLT is a language for transforming XML documents into other XML documents.

XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.

XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformation that are needed when XSLT is used as part of XSL.

A transformation expressed in XSLT describes rules for transforming a source tree into a result tree. The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree. A template is instantiated to create part of the result tree. The result tree is separate from the source tree. The structure of the result tree can be completely different from the structure of the source tree. In constructing the result tree, elements from the source tree can be filtered and reordered, and arbitrary structure can be added.

A transformation expressed in XSLT is called a stylesheet. This is because, in the case when XSLT is transforming into the XSL formatting vocabulary, the transformation functions as a stylesheet.

A stylesheet contains a set of template rules. A template rule has two parts: a pattern which is matched against nodes in the source tree and a template which can be instantiated to form part of the result tree. This allows a stylesheet to be applicable to a wide class of documents that have similar source tree structures.

A template is instantiated for a particular source element to create part of the result tree. A template can contain elements that specify literal result element structure. A template can also contain elements from the XSLT namespace that are instructions for creating result tree fragments. When a template is instantiated, each instruction is executed and replaced by the result tree fragment that it creates. Instructions can select and process descendant source elements. Processing a descendant element creates a result tree fragment by finding the applicable template rule and instantiating its template. Note that elements are only processed when they have



been selected by the execution of an instruction. The result tree is constructed by finding the template rule for the root node and instantiating its template.

A software module called an XSL processor is used to read XML documents and transform them into other XML documents with different styles.

The C implementation of the XSL processor followed the XSL Transformations standard (version 1.0, November 16, 1999) and included the required behavior of an XSL processor as specified in the XSLT specification.

---

## Functions

```
xslprocess(xmlctx *docctx, xmlctx *xslctx, xmlctx *resctx, xmlnode **result)
```

Processes XSL Stylesheet with XML document source and returns success or an error code.

## Function Prototypes

### xslprocess

#### Purpose

This function processes an XSL Stylesheet with an XML document source.

#### Syntax

```
uword xslprocess(xmlctx *docctx, xmlctx *xslctx,  
                xmlctx *resctx, xmlnode **result);
```

#### Parameters

xmlctx (IN/OUT) - The XML document context  
xslctx (IN) - The XSL stylesheet context  
resctx (IN) - The result document fragment context  
result (IN/OUT) - The result document fragment node

## W3C SAX APIs

SAX is a standard interface for event-based XML parsing, developed collaboratively by the members of the XML-DEV mailing list.

There are two major types of XML (or SGML) APIs:

- tree-based APIs, and
- event-based APIs.

A tree-based API compiles an XML document into an internal tree structure, then allows an application to navigate that tree using the Document Object Model (DOM), a standard tree-based API for XML and HTML documents.

An event-based API, on the other hand, reports parsing events (such as the start and end of elements) directly to the application through callbacks, and does not usually build an internal tree. The application implements handlers to deal with the different events, much like handling events in a graphical user interface.

Tree-based APIs are useful for a wide range of applications, but they often put a great strain on system resources, especially if the document is large (under very controlled circumstances, it is possible to construct the tree in a lazy fashion to

avoid some of this problem). Furthermore, some applications need to build their own, different data trees, and it is very inefficient to build a tree of parse nodes, only to map it onto a new tree.

In both of these cases, an event-based API provides a simpler, lower-level access to an XML document: you can parse documents much larger than your available system memory, and you can construct your own data structures using your callback event handlers.

To use SAX, an `xmlsaxcb` structure is initialized with function pointers and passed to the `xmlinit` call. A pointer to a user-defined context structure may also be included; that context pointer will be passed to each SAX function.

The SAX callback structure:

```
typedef struct
{
    sword (*startDocument)(void *ctx);
    sword (*endDocument)(void *ctx);
    sword (*startElement)(void *ctx, const oratext *name,
                          const struct xmlarray *attrs);
    sword (*endElement)(void *ctx, const oratext *name);
    sword (*characters)(void *ctx, const oratext *ch, size_t len);
    sword (*ignorableWhitespace)(void *ctx, const oratext *ch, size_t len);
    sword (*processingInstruction)(void *ctx, const oratext *target,
                                   const oratext *data);
    sword (*notationDecl)(void *ctx, const oratext *name,
                          const oratext *publicId, const oratext *systemId);
    sword (*unparsedEntityDecl)(void *ctx, const oratext *name,
                                const oratext *publicId,
                                const oratext *systemId,
                                const oratext *notationName);
    sword (*nsStartElement)(void *ctx, const oratext *qname,
                            const oratext *local, const oratext *nsp,
                            const struct xmlnodes *attrs);
} xmlsaxcb;
```

## Data Structures and Types

Callback Functions conforming to the SAX standard:

`sword (*characters)(void *ctx, const oratext *ch, size_t len)`

Receive notification of character data inside an element.

`sword (*endDocument)(void *ctx)`

Receive notification of the end of the document.

`sword (*endElement)(void *ctx, const oratext *name)`

Receive notification of the end of an element.

`sword (*ignorableWhitespace)(void *ctx, const oratext *ch, size_t len)`

Receive notification of ignorable whitespace in element content.

`sword (*notationDecl)(void *ctx, const oratext *name,  
const oratext *publicId, const oratext *systemId)`

Receive notification of a notation declaration.

`sword (*processingInstruction)(void *ctx, const oratext *target,  
const oratext *data)`

Receive notification of a processing instruction.

`sword (*startDocument)(void *ctx)`

Receive notification of the beginning of the document.

`sword (*startElement)(void *ctx, const oratext *name,  
const struct xmlattrs *attrs)`

Receive notification of the start of an element.

`sword (*unparsedEntityDecl)(void *ctx, const oratext *name,  
const oratext *publicId, const oratext *systemId,  
const oratext *notationName)`

Receive notification of an unparsed entity declaration.

---

## Non-SAX Callback Functions

```
sword (*nsStartElement)(void *ctx, const oratext *qname, const oratext *local,  
                        const oratext *namespace, const struct xmlattrs *attrs)
```

Receive notification of the start of a namespace for an element.

## Function Prototypes

---

### characters

#### Purpose

This callback function receives notification of character data inside an element.

#### Syntax

```
sword (*characters)(void *ctx, const oratext *ch, size_t len);
```

#### Parameters

ctx (IN) - client context pointer

ch (IN) - the characters

len (IN) - number of characters to use from the character pointer

#### Comments

### endDocument

#### Purpose

This callback function receives notification of the end of the document.

#### Syntax

```
sword (*endDocument)(void *ctx);
```

#### Parameters

ctx (IN) - client context

## Comments

### endElement

#### Purpose

This callback function receives notification of the end of an element.

#### Syntax

```
sword (*endElement)(void *ctx, const oratext *name);
```

#### Parameters

ctx (IN) - client context

name (IN) - element type name

## Comments

### ignorableWhitespace

#### Purpose

This callback function receives notification of ignorable whitespace in element content.

#### Syntax

```
sword (*ignorableWhitespace)(void *ctx, const oratext *ch, size_t len);
```

#### Parameters

ctx (IN) - client context

ch (IN) - whitespace characters

len (IN) - number of characters to use from the character pointer

## Comments

### notationDecl

Purpose



This callback function receives notification of a notation declaration.

**Syntax**

```
sword (*notationDecl)(void *ctx, const oratext *name, const oratext *publicId,  
                      const oratext *systemId);
```

**Parameters**

ctx (IN) - client context

name (IN) - notation name

publicId (IN) - notation public identifier, or null if not available

systemId (IN) - notation system identifier

**Comments****processingInstruction****Purpose**

This callback function receives notification of a processing instruction.

**Syntax**

```
sword (*processingInstruction)(void *ctx, const oratext *target,  
                               const oratext *data);
```

**Parameters**

ctx (IN) - client context

target (IN) - processing instruction target

data (IN) - processing instruction data, or null if none is supplied

## Comments

### startDocument

#### Purpose

This callback function receives notification of the beginning of the document.

#### Syntax

```
sword (*startDocument)(void *ctx);
```

#### Parameters

ctx (IN) - client context

## Comments

### startElement

#### Purpose

This callback function receives notification of the beginning of an element.

#### Syntax

```
sword (*startElement)(void *ctx, const oratext *name,  
                      const struct xmlattrs *attrs);
```

#### Parameters

ctx (IN) - client context

name (IN) - element type name

attrs (IN) - specified or defaulted attributes

## Comments

### unparsedEntityDecl

#### Purpose

This callback function receives notification of an unparsed entity declaration.

#### Syntax

```
sword (*unparsedEntityDecl)(void *ctx, const oratext *name,  
                             const oratext *publicId, const oratext *systemId,  
                             const oratext *notationName);
```

#### Parameters

ctx (IN) - client context

name (IN) - entity name

publicId (IN) - entity public identifier, or null if not available

systemId (IN) - entity system identifier

notationName (IN) - name of the associated notation

## Comments

### nsStartElement

#### Purpose

This callback function receives notification of the start of a namespace for an element.

#### Syntax

```
sword (*nsStartElement)(void *ctx, const oratext *qname, const oratext *local,  
                        const oratext *namespace, const struct xmlattrs *attrs);
```

#### Parameters

ctx (IN) - client context

qname (IN) - element fully qualified name

local (IN) - element local name

namespace (IN) - element namespace (URI)

attrs (IN) - specified or defaulted attributes

### Comments

## W3C DOM APIs

The *Document Object Model* (**DOM**) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term *document* is used in the broad sense -- increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the DOM, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the DOM, with a few exceptions -- in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

One important objective of the W3C specification for the DOM is to provide a standard programming interface that can be used in a wide variety of environments and applications. The DOM is designed to be used with any programming language. Since the DOM standard is object-oriented, for the C adaptation, some changes had to be made:

- Reused function names had to be expanded, e.g. `getValue` in the attribute class is given the unique name `getAttrValue`, matching the pattern established by `getNodeValue`.
- Also, some functions were added to extend the DOM. For example, there is no function defined which returns the number of children of a node, so `numChildNodes` was invented, etc.

The implementation of this C DOM interface follows REC-DOM-Level-1-19981001.

## DOM Functions

appendChild	Node::appendChild	Append child node to current node
appendData	Node::appendData	Append character data to end of node's current data
cloneNode		Create a new node identical to the current one
createAttribute		Create a new attribute for an element node
createCDATASection		Create a CDATA node
createComment		Create a comment node
createDocumentFragment		Create a document fragment node
createElement		Create an element node
createEntityReference		Create an entity reference node
createProcessingInstruction		Create a processing instruction (PI) node
createTextNode		Create a text node
deleteData		Remove substring from a node's character data
getAttributeName		Return an attribute's name
getAttributeSpecified		Return value of attribute's <i>specified</i> flag [DOM <code>getAttributeSpecified</code> ]
getAttributeValue		Return an attribute's value (definition) [DOM <code>getAttributeValue</code> ]
getAttribute		Return the value of an attribute
getAttributeIndex		Return an element's attribute given its index
getAttributeNode		Get an element's attribute node given its name [DOM <code>getAttributeName</code> ]
getAttributes		Return array of element's attributes

<code>getCharData</code>	Return character data for a TEXT node [DOM <code>getData</code> ]
<code>getCharLength</code>	Return length of TEXT node's character data [DOM <code>getLength</code> ]
<code>getChildNode</code>	Return indexed node from array of nodes [DOM item]
<code>getChildNodes</code>	Return array of node's children
<code>getContentModel</code>	Returns the content model for an element from the DTD [DOM extension]
<code>getDocument</code>	Return top-level DOCUMENT node [DOM extension]
<code>getDocumentElement</code>	Return highest-level (root) ELEMENT node
<code>getDocType</code>	Return current DTD
<code>getDocTypeEntities</code>	Return array of DTD's general entities
<code>getDocTypeName</code>	Return name of DTD
<code>getDocTypeNotations</code>	Return array of DTD's notations
<code>getElementsByTagName</code>	Return list of elements with matching name
<code>getEntityNotation</code>	Return an entity's NDATA [DOM <code>getNotation</code> ]
<code>getEntityPubID</code>	Return an entity's public ID [DOM <code>getPublicId</code> ]
<code>getEntitySysID</code>	Return an entity's system ID [DOM <code>getSystemId</code> ]
<code>getFirstChild</code>	Return the first child of a node
<code>getImplementation</code>	Return DOM-implementation structure (if defined)
<code>getLastChild</code>	Return the last child of a node
<code>getNextSibling</code>	Return a node's next sibling
<code>getNamedItem</code>	Returns the named node from a list of nodes

---

<code>getNodeMapLength</code>	Returns number of entries in a NodeMap [DOM <code>getLength</code> ]
<code>getNodeName</code>	Returns a node's name
<code>getNodeType</code>	Returns a node's type code (enumeration)
<code>getNodeValue</code>	Returns a node's "value", its character data
<code>getNotationPubID</code>	Returns a notation's public ID [DOM <code>getPublicId</code> ]
<code>getNotationSysID</code>	Returns a notation's system ID [DOM <code>getSystemId</code> ]
<code>getOwnerDocument</code>	Returns the DOCUMENT node containing the given node
<code>getPIData</code>	Returns a processing instruction's data [DOM <code>getData</code> ]
<code>getPITarget</code>	Returns a processing instruction's target [DOM <code>getTarget</code> ]
<code>getParentNode</code>	Returns a node's parent node
<code>getPreviousSibling</code>	Returns a node's "previous" sibling
<code>getTagName</code>	Returns a node's "tagname", same as name for now
<code>hasAttributes</code>	Determine if element node has attributes [DOM extension]
<code>hasChildNodes</code>	Determine if node has children
<code>hasFeature</code>	Determine if DOM implementation supports a specific feature
<code>insertBefore</code>	Inserts a new child node before the given reference node
<code>insertData</code>	Inserts new character data into a node's existing data
<code>isStandalone</code>	Determine if document is standalone [DOM extension]
<code>nodeValid</code>	Validate a node against the current DTD [DOM extension]

<code>normalize</code>	Normalize a node by merging adjacent TEXT nodes
<code>numAttributes</code>	Returns number of element node's attributes [DOM extension]
<code>numChildNodes</code>	Returns number of node's children [DOM extension]
<code>removeAttribute</code>	Removes an element's attribute given its names
<code>removeAttributeNode</code>	Removes an element's attribute given its pointer
<code>removeChild</code>	Removes a node from its parents list of children
<code>removeNamedItem</code>	Removes a node from a list of nodes given its name
<code>replaceChild</code>	Replace one node with another
<code>replaceData</code>	Replace a substring of a node's character data with another string
<code>setAttribute</code>	Sets (adds or replaces) a new attribute for an element node given the attribute's name and value
<code>setAttributeNode</code>	Sets (adds or replaces) a new attribute for an element node given a pointer to the new attribute
<code>setNamedItem</code>	Sets (adds or replaces) a new node in a parent's list of children
<code>setNodeValue</code>	Sets a node's "value" (character data)
<code>setPIData</code>	Sets a processing instruction's data [DOM setData]
<code>splitText</code>	Split a node's character data into two parts
<code>substringData</code>	Return a substring of a node's character data



## Function Prototypes

### appendChild

#### Purpose

Adds new node to the end of the list of children for the given parent and returns the node added.

#### C Prototype

```
xmlnode *appendChild(xmlctx *ctx, xmlnode *parent, xmlnode *newnode)
```

#### C++ Prototype

```
Node* Node::appendChild(Node *newChild)
```

#### Parameters

ctx	(IN)	XML context
parent	(IN)	parent node
newnode	(IN)	new node to append

#### C Example

```
xmlnode *node, *parent;  
...  
  
if (node = createElement(ctx, "node"))  
    appendChild(ctx, parent, node);
```

### appendData

#### Purpose

Append the given string to the character data of a TEXT or CDATA node.

## Syntax

```
void appendData(xmlctx *ctx, xmlnode *node, const oratext *arg)
```

## Parameters

ctx	(IN)	XML context
node	(IN)	pointer to node
arg	(IN)	new data to append

## Example

```
xmlnode *node;  
...  
getNodeValue(node) -> "foo"  
appendData(ctx, node, "bar");  
getNodeValue(node) -> "foobar"
```

## cloneNode

### Purpose

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (`parentNode` returns `NULL`).

Cloning an Element copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child Text node. Cloning any other type of node simply returns a copy of this node.

A *deep* clone differs in that the node's children are also recursively cloned instead of just pointed-to.

### Syntax

```
xmlnode *cloneNode(xmlctx *ctx, const xmlnode *old, boolean deep)
```

## Parameters

ctx	(IN)	XML context
old	(IN)	old node to clone
deep	(IN)	recursion flag

## createAttribute

### Purpose

Create a new ATTRIBUTE node with the given name and value. The new node is unattached and must be added to an element node with `setAttributeNode`.

### Syntax

```
xmlnode *createAttribute(xmlctx *ctx, const oratext *name, const oratext *value)
```

## Parameters

ctx	(IN)	XML context
name	(IN)	name of new attribute
value	(IN)	value of new attribute

### Example

```
xmlnode *attr, *elem;
...
if (attr = createAttribute(ctx, "attr1", "value1"))
{
    setAttributeNode(ctx, elem, attr, NULL);
}
```

## createCDATASection

### Purpose

Create a new CDATA node.

## Syntax

```
xmlnode *createCDATASection(xmlctx *ctx, const oratext *data)
```

## Parameters

ctx (IN) XML context

data (IN) CDATA body

## Example

```
xmlnode *node, *parent;
...
if (node = createCDATASection(ctx, "<greeting>H'oi!</greeting>"))
    appendChild(ctx, parent, node);
```

## createComment

### Purpose

Create a new COMMENT node.

### Syntax

```
xmlnode *createComment(xmlctx *ctx, const oratext *data)
```

### Parameters

ctx (IN) XML context

data (IN) text of comment

### Example

```
xmlnode *node, *parent;
...
if (node = createComment(ctx, "From here on this document is unfinished"))
    appendChild(ctx, parent, node);
```

## createDocumentFragment

### Purpose

Create a new DOCUMENT\_FRAGMENT node. A document fragment is a lightweight document object that contains one or more children, but does not have the overhead of a full document. It can be used in some operations (inserting for example) in place of a simple node, in which case all the fragment's children are operated on instead of the fragment node itself.

### Syntax

```
xmlnode *createDocumentFragment(xmlctx *ctx)
```

### Parameters

ctx        (IN)     XML context

### Example

```
xmlnode *frag, *fragelem, *fragtext;
...
if ((frag = createDocumentFragment(ctx)) &&
    (fragelem = createElement(ctx, (oratext *) "FragElem")) &&
    (fragtext = createTextNode(ctx, (oratext *) "FragText")))
{
    appendChild(ctx, frag, fragelem);
    appendChild(ctx, frag, fragtext);
}
```

## createElement

### Purpose

Create a new ELEMENT node.

### Syntax

```
xmlnode *createElement(xmlctx *ctx, const oratext *elname)
```

## Parameters

ctx (IN) XML context  
elname (IN) name of new element

## Example

```
xmlnode *node, *parent;  
...  
if (node = createElement(ctx, "BOOK"))  
    appendChild(ctx, parent, node);
```

## createEntityReference

### Purpose

Create a new ENTITY\_REFERENCE node.

### Syntax

```
xmlnode *createEntityReference(xmlctx *ctx, const oratext *name)
```

## Parameters

ctx (IN) XML context  
name (IN) name of entity to reference

## Example

```
xmlnode *node, *parent;  
...  
if (node = createEntityReference(ctx, "homephone"))  
    appendChild(ctx, parent, node);
```

## createProcessingInstruction

### Purpose

Create a new PROCESSING\_INSTRUCTION node with the given target and contents.

### Syntax

```
xmlnode *createProcessingInstruction(xmlctx *ctx, const oratext *target,  
                                   const oratext *data)
```

### Parameters

ctx	(IN)	XML context
target	(IN)	PI target
data	(IN)	PI definition

### Example

```
xmlnode *node, *parent;  
...  
if (node = createProcessingInstruction(ctx, "target", "definition"))  
    appendChild(ctx, parent, node);
```

## createTextNode

### Purpose

Create a new TEXT node with the given contents.

### Syntax

```
xmlnode *createTextNode(xmlctx *ctx, const oratext *data)
```

## Parameters

ctx (IN) XML context  
data (IN) data for node

## Example

```
xmlnode *node, *parent;  
...  
if (node = createTextNode(ctx, "riverrun, past Eve and Adam's..."))  
    appendChild(ctx, parent, node);
```

## deleteData

### Purpose

Delete a substring from the node's character data.

### Syntax

```
void deleteData(xmlctx *ctx, xmlnode *node, ub4 offset, ub4 count)
```

## Parameters

ctx (IN) XML context  
node (IN) pointer to node  
offset (IN) offset of start of substring (0 is first char)  
count (IN) length of substring

## Example

```
xmlnode *node;  
...  
getNodeValue(node) -> "phoenix"  
deleteData(ctx, node, 2, 1);  
getNodeValue(node) -> "phenix"
```



## getAttribute

### Purpose

Returns one attribute from an array of attributes, given an index (starting at 0). Fetch the attribute name and/or value (with `getAttrName` and `getAttrValue`). On error, returns `NULL`.

### Syntax

```
const oratext *getAttribute(const xmlnode *node, const oratext *name)
```

### Parameters

<code>node</code>	(IN)	node whose attributes to scan
<code>name</code>	(IN)	name of the attribute

### Example

```
xmlnode *node, *attr;  
xmlnodes *nodes;  
const oratext *attrval;  
...  
if (nodes = getAttributes(node))  
{  
    attr = getAttributeIndex(nodes, 1); /* second attribute */  
    attrval = getAttribute(attr, "foo");  
    ...  
}
```

## getAttributeIndex

### Purpose

Returns one attribute from an array of attributes, given an index (starting at 0). Fetch the attribute name and/or value (with `getAttrName` and `getAttrValue`). On error, returns `NULL`.

### Syntax

```
xmlnode *getAttributeIndex(const xmlnodes *attrs, size_t index)
```

## Parameters

`attrs` (IN) pointer to attribute nodes structure (as returned by `getAttributes`)

`index` (IN) zero-based attribute# to return

## Example

```
xmlnode *node, *attr;
xmlnodes *nodes;
...
if (nodes = getAttributes(node))
{
    attr = getAttributeIndex(nodes, 1);    /* second attribute */
    ...
}
```

## getAttributeNode

### Purpose

Returns a pointer to the element node's attribute of the given name. If no such thing exists, returns `NULL`.

### Syntax

```
xmlnode *getAttributeNode(const xmlnode *elem, const oratext *name)
```

## Parameters

`elem` (IN) pointer to element node

`name` (IN) name of attribute

## Example

```
xmlnode *node, *attr;
...
if (attr = getAttributeNode(elem, "attr1"))
    ...
```

## getAttributes

### Purpose

Returns an array of all attributes of the given node. This pointer may then be passed to `getAttribute` to fetch individual attribute pointers, or to `numAttributes` to return the total number of attributes. If no attributes are defined, returns `NULL`.

### Syntax

```
xmlnodes *getAttributes(const xmlnode *node)
```

### Parameters

node (IN) node whose attributes to return

### Example

```
xmlnode *node;  
xmlnodes *nodes;  
...  
if (nodes = getAttributes(node))  
    ...
```

## getAttrName

### Purpose

Given a pointer to an attribute, returns the name of the attribute. Under the DOM spec, this is a method named `getName`.

### Syntax

```
const oratext *getAttrName(const xmlnode *attr)
```

## Parameters

`attr` (IN) pointer to attribute (see `getAttribute`)

## Example

```
xmlnode *elem, *attr;
...
attr = setAttribute(ctx, elem, "x", "y");
getAttribute(attr) -> "x"
```

## getAttributeSpecified

### Purpose

Return the 'specified' flag for the attribute: if this attribute was explicitly given a value in the original document or through the DOM, this is `TRUE`; otherwise, it is `FALSE`. If the node is not an attribute, returns `FALSE`. Under the DOM spec, this is a method named `getSpecified`.

### Syntax

```
boolean getAttributeSpecified(const xmlnode *attr)
```

## Parameters

`attr` (IN) pointer to attribute (see `getAttribute`)

## Example

```
xmlnode *elem, *attr;
...
attr = setAttribute(ctx, elem, "x", "y");
getAttributeSpecified(attr) -> TRUE
```

## getAttrValue

### Purpose

Given a pointer to an attribute, returns the "value" (definition) of the attribute. Under the DOM spec, this is a method named `getValue`.

### Syntax

```
const oratext *getAttrValue(const xmlnode *attr)
```

### Parameters

`attr` (IN) pointer to attribute (see `getAttribute`)

### Example

```
xmlnode *elem, *attr;
...
attr = setAttribute(ctx, elem, "x", "y");
getAttrValue(attr) -> "y"
```

## getCharData

### Purpose

Returns the character data of a TEXT or CDATA node. Under the DOM spec, this is a method named `getData`.

### Syntax

```
const oratext *getCharData(const xmlnode *node)
```

### Parameters

`node` (IN) pointer to text node

### Example

```
xmlnode *node;
...
if (node = createTextNode(ctx, "riverrun"))
```

```
getCharData(node) -> "riverrun"
```

## getCharLength

### Purpose

Returns the length of the character data of a TEXT or CDATA node. Under the DOM spec, this is a method named `getLength`.

### Syntax

```
ub4 getCharLength(const xmlnode *node)
```

### Parameters

node (IN) pointer to text node

### Example

```
xmlnode *node;  
...  
if (node = createTextNode(ctx, "promptly"))  
    getCharLength(node) -> 8
```

## getChildNode

### Purpose

Returns the *n*th node in an array of nodes, or NULL if the numbered node does not exist. Invented function, not in DOM, but named to match the DOM pattern.

### Syntax

```
xmlnode* getChildNode(const xmlnodes *nodes, size_t index)
```

## Parameters

`nodes` (IN) array of nodes (see `getChildNodes`)  
`index` (IN) zero-based child#

## Example

```
xmlnode *node, *child;  
xmlnodes *nodes;  
...  
if (nodes = getChildNodes(node))  
{  
    child = getChildNode(nodes, 1); /* second child node */  
    ...  
}
```

## getChildNodes

### Purpose

Returns the array of children of the given node. This pointer may then be passed to `getChildNode` to fetch individual children.

### Syntax

```
xmlnodes* getChildNodes(const xmlnode *node)
```

### Parameters

`node` (IN) node whose children to return

### Example

```
xmlnode *node;  
xmlnodes *nodes;  
...  
if (nodes = getChildNodes(node))  
    ...
```

## getContentModel

### Purpose

Returns the content model for the named element from the current DTD. The content model is composed of `xmlnodes`, so may be traversed with the same functions as the parsed document. See also the `getModifier` function which returns the '?', '\*', and '+' modifiers to content model nodes.

### Syntax

```
xmlnode *LpxGetContentModel(xmltdt *dtd, oratext *name)
```

### Parameters

`dtd` (IN) pointer to the DTD  
`name` (IN) name of element

## getDocType

### Purpose

Returns a pointer to the (opaque) DTD for the current document.

### Syntax

```
xmltdt* getDocType(xmlctx *ctx)
```

### Parameters

`ctx` (IN) XML parser context

### Example

```
xmlnodes *nodes;  
...  
nodes = getDocTypeEntities(getDocType(ctx));
```



## getDocTypeEntities

### Purpose

Returns an array of (general) entities defined for the given DTD.

### Syntax

```
xmlnodes *getDocTypeEntities(xmltdt* dtd)
```

### Parameters

dtd (IN) pointer to DTD

### Example

```
xmltdt *dtd;  
xmlnodes *entities;  
...  
dtd = getDocType(ctx);  
entities = getDocTypeEntities(dtd);
```

## getDocTypeName

### Purpose

Returns the given DTD's name.

### Syntax

```
oratext *getDocTypeName(xmltdt* dtd)
```

## Parameters

dtd (IN) pointer to DTD

## getDocTypeNotations

### Purpose

Returns an array of notations defined for the given DTD.

### Syntax

```
xmlnodes *getDocTypeNotations(xmlDTD* dtd)
```

## Parameters

dtd (IN) pointer to DTD

## Example

```
xmlDTD *dtd;  
xmlnodes *notations;  
...  
dtd = getDocType(ctx);  
notations = getDocTypeNotations(dtd);
```

## getElementsByTagName

### Purpose

Returns a list of all elements (within the tree rooted at the given node) with a given tag name in the order in which they would be encountered in a pre-order traversal of the tree. If root is NULL, the entire document is searched. The special value "\*" matches all tags.

### Syntax

```
xmlnodes *getElementsByTagName(xmlctx *ctx, xmlnode *root, const oratext *name)
```

## Parameters

ctx (IN) XML parser context  
root (IN) root node of tree  
name (IN) element tag name

## Example

```
xmlnodes *nodes;  
...  
nodes = getElementByTagName(ctx, NULL, "ACT"); /* find all ACT elements */
```

## getDocument

### Purpose

Returns the root node of the parsed document. The root node is always of type DOCUMENT\_NODE. Compare to the `getDocumentElement` function, which returns the root *element* node, which is a child of the DOCUMENT node.

### Syntax

```
xmlnode* getDocument(xmlctx *ctx)
```

#### Parameters

ctx (IN) XML parser context

## getDocumentElement

### Purpose

Returns the root element (node) of the parsed document. The entire document is rooted at this node. Compare to `getDocument` which returns the uppermost DOCUMENT node (the parent of the root element node).

### Syntax

```
xmlnode* getDocumentElement(xmlctx *ctx)
```

## Parameters

ctx (IN) XML parser context

## getEntityNotation

### Purpose

Returns an entity node's NDATA (notation). Under the DOM spec, this is a method named `getNotationName`.

### Syntax

```
const oratext *getEntityNotation(const xmlnode *ent)
```

## Parameters

ent (IN) pointer to entity

## Example

```
<!NOTATION n SYSTEM "http://www.w3.org/">
<!ENTITY e SYSTEM "http://www.w3.org/" NDATA n>

xmlnode *ent; /* assume ent will be set to ENTITY node above */
...
getEntityNotation(ent) -> "n"
```

## getEntityPubID

### Purpose

Returns an entity node's public ID. Under the DOM spec, this is a method named `getPublicId`.

### Syntax

```
const oratext *getEntityPubID(const xmlnode *ent)
```

## Parameters

ent (IN) pointer to entity

## Example

```
<!ENTITY e PUBLIC "PublicID" "nop.ent">

xmlnode *ent; /* assume ent will be set to ENTITY node above */
...
getEntityPubID(ent) -> "PublicID"
```

## getEntitySysID

### Purpose

Returns an entity node's system ID. Under the DOM spec, this is a method named `getSystemId`.

### Syntax

```
const oratext *getEntitySysID(const xmlnode *ent)
```

## Parameters

ent (IN) pointer to entity

## Example

```
<!ENTITY e PUBLIC "PublicID" "nop.ent">

xmlnode *ent; /* assume ent will be set to ENTITY node above */
...
getEntitySysID(ent) -> "nop.ent"
```

## getFirstChild

### Purpose

Returns the first child of the given node, or `NULL` if the node has no children.

### Syntax

```
xmlnode* getFirstChild(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

### Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *elem; /* assume elem will point to element Thing */
```

```
...
```

```
getFirstChild(elem) -> element "A"
```

## getImplementation

### Purpose

This function returns a pointer to the `DOMImplementation` structure for this implementation, or `NULL` if no such information is available.

### Syntax

```
xmlDOMimp* getImplementation(xmlctx *ctx)
```

### Parameters

ctx (IN) XML context

## getLastChild

### Purpose

Returns the last child of the given node, or `NULL` if the node has no children.

## Syntax

```
xmlnode* getLastChild(const xmlnode *node)
```

## Parameters

node (IN) pointer to node

## Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *elem; /* assume elem will point to element Thing */
...
getLastChild(elem) -> element "C"
```

## getNamedItem

### Purpose

Returns the named node from an array nodes; sets the user's index (if provided) to the child# of the node (first node is zero).

### Syntax

```
xmlnode *getNamedItem(const xmlnodes *nodes, const oratext *name, size_t *index)
```

### Parameters

nodes (IN) array of nodes  
name (IN) name of node to fetch  
index (OUT) index of found node

### Example

```
xmlnode *node, *elem;
xmlnodes *nodes;
size_t index;
...
if (nodes = getChildNodes(elem))
{
```

```
        node = getNamedItem(nodes, "FOO", &index);
        ...
    }
```

## getNextSibling

### Purpose

This function returns a pointer to the next sibling of the given node, that is, the next child of the parent. For the last child, `NULL` is returned.

### Syntax

```
xmlnode* getNextSibling(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

### Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *node, *elem; /* assume elem will point to node Thing */
...
for (node = getFirstChild(elem); node; node = getNextSibling(node))
    ...node will be A then B then C...
```

## getNodeMapLength

### Purpose

Given an array of nodes (as returned by `getChildNodes`), returns the number of nodes in the map. Under the DOM spec, this is a member function named `getLength`.

### Syntax

```
size_t getNodeMapLength(const xmlnodes *nodes)
```



## Parameters

nodes (IN) array of nodes

## Example

```
<Thing><A/><B/><C/></Thing>

xmlnodes *nodes;
xmlnode *elem; /* assume elem will point to node Thing */
...
if (nodes = getChildNodes(elem))
    getNodeMapLength(nodes) -> 3
```

## getNodeName

### Purpose

Returns the name of the given node, or NULL if the node has no name. Note that "tagname" and "name" are currently synonymous.

### Syntax

```
const oratext* getNodeName(const xmlnode *node)
```

## Parameters

node (IN) pointer to node

## Example

```
<Thing><A/><B/><C/></Thing>

xmlnode *elem; /* assume elem will point to node Thing */
...
getNodeName(elem) -> "Thing"
```

## getNodeType

### Purpose

Returns the type code for a node.

### Syntax

```
xmlIntType getNodeType(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

### Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *elem; /* assume elem will point to node Thing */  
...  
getNodeType(elem) -> ELEMENT_NODE
```

## getNodeValue

### Purpose

Returns the "value" (associated character data) for a node, or NULL if the node has no data.

### Syntax

```
const oratext* getNodeValue(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

### Example

```
<!--This is a comment-->
```

```
xmlnode *node; /* assume node will point to comment node above */  
...  
getNodeValue(node) -> "This is a comment"
```

## getNotationPubID

### Purpose

Return a notation node's public ID. Under the DOM spec, this is a method named `getPublicId`.

### Syntax

```
const oratext *getNotationPubID(const xmlnode *note)
```

### Parameters

`note` (IN) pointer to node

### Example

```
<!NOTATION n PUBLIC "whatever">
```

```
xmlnode *note; /* assume note will point to notation node above */  
...  
getNotationPubID(note) -> "whatever"
```

## getNotationSysID

### Purpose

Return a notation node's system ID. Under the DOM spec, this is a method named `getSystemId`.

### Syntax

```
const oratext *getNotationSysID(const xmlnode *note)
```

### Parameters

`note` (IN) pointer to node

## Example

```
<!NOTATION n SYSTEM "http://www.w3.org/">

xmlnode *note; /* assume note will point to notation node above */
...
getNotationSysID(note) -> "http://www.w3.org/"
```

## getOwnerDocument

### Purpose

Returns the document node which contains the given node. An XML document is always rooted in a node of type `DOCUMENT_NODE`. Calling `getOwnerDocument` on any node in the document returns that document node.

### Syntax

```
xmlnode* getOwnerDocument(xmlnode *node)
```

### Parameters

node (IN) pointer to node

## getParentNode

### Purpose

Returns the parent node of the given node. For the top-most node, `NULL` is returned.

### Syntax

```
xmlnode* getParentNode(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

### Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *elem; /* assume elem will point to node A */  
...  
getParentNode(elem) -> node Thing
```

## getPIData

### Purpose

Returns a Processing Instruction's (PI) data string. Under the DOM spec, this is a method named `getData`.

### Syntax

```
const oratext *getPIData(const xmlnode *pi)
```

### Parameters

`pi` (IN) pointer to PI node

### Example

```
<?PI Blither blather?>
```

```
xmlnode *pi; /* assume pi will point to PI node above */  
...  
getPIData(pi) -> "Blither blather"
```

## getPITarget

### Purpose

Returns a Processing Instruction's (PI) target string. Under the DOM spec, this is a method named `getTarget`.

### Syntax

```
const oratext *getPITarget(const xmlnode *pi)
```

## Parameters

`pi` (IN) pointer to PI node

## Example

```
<?PI Blither blather?>

xmlnode *pi; /* assume pi will point to PI node above */
...
getPITarget(pi) -> "PI"
```

## getPreviousSibling

### Purpose

Returns the previous sibling of the given node. That is, the node at the same level which came before this one. For the first child of a node, `NULL` is returned.

### Syntax

```
xmlnode* getPreviousSibling(const xmlnode *node)
```

## Parameters

`node` (IN) pointer to node

## Example

```
<Thing><A/><B/><C/></Thing>

xmlnode *node, *elem; /* assume elem will point to node Thing */
...
for (node = getLastChild(elem); node; node = getPreviousSibling(node))
    ...node will be C then B then A...
```

## getElementsByTagName

### Purpose

Returns the "tagname" of a node, which is the same as its name for now, see `getNodeName`. The DOM says "...even though there is a generic `nodeName` attribute on the Node interface, there is still a `tagName` attribute on the Element interface; these two attributes must contain the same value, but the Working Group considers it worthwhile to support both, given the different constituencies the DOM API must satisfy.

### Syntax

```
const oratext *getElementsByTagName(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

## hasAttributes

### Purpose

Determines if the given node has any defined attributes, returning `TRUE` if so, `FALSE` if not. This is a DOM extension named after the pattern started by `hasChildNodes`.

### Syntax

```
boolean hasAttributes(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

## hasChildNodes

### Purpose

Determines if the given node has children, returning `TRUE` if so, `FALSE` if not. The same result can be achieved by testing if `getChildNodes` returns a pointer (has children) or `NULL` (no children).

### Syntax

```
boolean hasChildNodes(const xmlnode *node)
```

### Parameters

node (IN) pointer to node

## hasFeature

### Purpose

Tests if the DOM implementation implements a specific feature and version. *feature* is the package name of the feature to test. In DOM Level 1, the legal values are "HTML" and "XML" (case-insensitive). *version* is the version number of the package name to test. In DOM Level 1, this is the string "1.0". If the version is not specified, supporting any version of the feature will cause the method to return `TRUE`.

### Syntax

```
boolean hasFeature(xmlctx *ctx, const oratext *feature, const oratext *version)
```

### Parameters

ctx (IN) XML context  
feature (IN) the package name of the feature to test  
version (IN) the version number of the package name to test



## insertBefore

### Purpose

Inserts a new node into the given parent node's list of children before the existing reference node. If the reference node is `NULL`, appends the new node at the end of the list. If the new node is a `DocumentFragment`, its children are inserted, in the same order, instead of the fragment itself. If the new node is already in the tree, it is first removed.

### Syntax

```
xmlnode *insertBefore(xmlctx *ctx, xmlnode *parent,  
                      xmlnode *newChild, xmlnode *refChild)
```

### Parameters

<code>ctx</code>	(IN)	XML context
<code>parent</code>	(IN)	parent node to insert into
<code>newChild</code>	(IN)	new child node to insert
<code>refChild</code>	(IN)	reference node to insert before

### Example

```
<Thing><A/><B/><C/></Thing>
```

```
xmlnode *elem, *new, *ref; /* assume elem points to Thing, new is a new  
                           element "Z", and ref points to node B */
```

```
...  
insertBefore(ctx, elem, new, ref);
```

```
<Thing><A/><Z/><B/><C/></Thing>
```

## insertData

### Purpose

Inserts a string into the node character data at the specified offset.

## Syntax

```
void insertData(xmlctx *ctx, xmlnode *node, ub4 offset, const oratext *arg)
```

## Parameters

ctx	(IN)	XML context
node	(IN)	pointer to node
offset	(IN)	insertion point (0 is first position)
refChild	(IN)	new string to insert

## Example

```
xmlnode *node;  
...  
getNodeValue(node) -> "abcdefg"  
insertData(ctx, node, 3, "ZZZ");  
getNodeValue(node) -> "abcZZZdefg"
```

## isStandalone

### Purpose

Returns the value of the `standalone` flag as specified in the document's `<?xml?>` processing instruction. This is an invented function, not in DOM spec, but named to match the DOM pattern.

### Syntax

```
boolean isStandalone(xmlctx *ctx)
```

### Parameters

ctx	(IN)	XML parser context
-----	------	--------------------

## nodeValid

### Purpose

Validate a node against the DTD. Returns 0 on success, else a non-zero error code (which can be looked up in the message file). This function is provided for

applications which construct their own documents via the API and/or Class Generator. Normally the parser will validate the document and the user need not call `nodeValid` explicitly.

### Syntax

```
uword nodeValid(xmlctx *ctx, const xmlnode *node)
```

### Parameters

ctx (IN) XML context  
node (IN) pointer to node

## normalize

### Purpose

"*Normalizes*" an element, i.e. merges adjacent TEXT nodes. Adjacent TEXT nodes don't happen during a normal parse, only when extra nodes are inserted via the DOM.

### Syntax

```
void normalize(xmlctx *ctx, xmlnode *elem)
```

### Parameters

ctx (IN) XML context  
elem (IN) pointer to element node

### Example

```
xmlnode *node, *t1, *t2;  
...  
if ((node = createElement(ctx, "FOO")) &&  
    (t1 = createTextNode(ctx, "one of ")) &&  
    (t2 = createTextNode(ctx, "these days")) &&  
    appendChild(ctx, node, t1) &&  
    appendChild(ctx, node, t2))  
{
```

```
<FOO>"one of " "these days"</FOO>
normalize(ctx, node);
<FOO>"one of these days"</FOO>
}
```

## numAttributes

### Purpose

Returns the number of defined attributes in an attribute array (as returned by `getAttributes`). This is an invented function, not in the DOM spec, but named after the DOM pattern.

### Syntax

```
size_t numAttributes(const xmlnodes *attrs)
```

### Parameters

`attrs` (IN) array of attributes

### Example

```
xmlnodes *nodes;
xmlnode *node;
size_t i;
...
if (nodes = getAttributes(node))
{
    for (i = 0; i < numAttributes(nodes); i++)
        ...
}
```

## numChildNodes

### Purpose

Returns the number of children in an array of nodes (as returned by `getChildNodes`). This is an invented function, not in the DOM spec, but named after the DOM pattern.

## Syntax

```
size_t numChildNodes(const xmlnodes *nodes)
```

## Parameters

nodes (IN) pointer to opaque nodes structure

## Example

```
xmlnodes *nodes;
xmlnode *elem;
size_t i;
...
if (nodes = getChildNodes(elem))
{
    for (i = 0; i < numChildNodes(nodes); i++)
        ...
}
```

## removeAttribute

### Purpose

Removes the named attribute from an element node. If the removed attribute has a default value it is immediately replaced.

### Syntax

```
void removeAttribute(xmlnode *elem, const oratext *name)
```

### Parameters

elem (IN) pointer to element node  
name (IN) name of attribute to remove

### Example

```
<!ATTLIST FOO attr CDATA 'default'>
```

```
xmlnode *elem; /* assume elem point to a FOO node */  
...  
<FOO attr="snark"/>  
removeAttribute(elem, "attr");  
<FOO attr="default"/>
```

## removeAttributeNode

### Purpose

Removes an attribute from an element, given a pointer to the attribute. If successful, returns the attribute node back. On error, returns `NULL`.

### Syntax

```
xmlnode *removeAttributeNode(xmlnode *elem, xmlnode *attr)
```

### Parameters

<code>elem</code>	(IN)	pointer to element node
<code>attr</code>	(IN)	attribute node to remove

### Example

```
xmlnode *elem, *attr;  
...  
if (attr = getAttributeNode(elem, "attr1"))  
    removeAttributeNode(elem, attr);
```

## removeChild

### Purpose

Removes the given node from its parent and returns it.

### Syntax

```
xmlnode *removeChild(xmlnode *node)
```

## Parameters

node (IN) old node to remove

## Example

```
xmlnodes *nodes;
xmlnode *elem, *node;
...
if ((nodes = getChildNodes(elem)) &&
    (node = getNamedItem(nodes, "B", NULL))
    {
    <Thing><A/><B/><C/></Thing>
    removeChild(node);
    <Thing><A/><C/></Thing>
    }
```

## removeNamedItem

### Purpose

Removes the named node from an array of nodes.

### Syntax

```
xmlnode *removeNamedItem(xmlnodes *nodes, const oratext *name)
```

## Parameters

nodes (IN) list of nodes

name (IN) name of node to remove

## Example

```
xmlnodes *nodes;
xmlnode *elem;
...
if (nodes = getChildNodes(elem))
{
    <Thing><A/><B/><C/></Thing>
```

```
removeNamedItem(nodes, "B");  
<Thing><A/><C/></Thing>  
}
```

## replaceChild

### Purpose

Replaces an existing child node with a new node and returns the old node. If the new node is already in the tree, it is first removed.

### Syntax

```
xmlnode *replaceChild(xmlctx *ctx, xmlnode *newChild, xmlnode *oldChild)
```

### Parameters

ctx	(IN)	XML context
newChild	(IN)	new replacement node
oldChild	(IN)	old node being replaced

### Example

```
xmlnodes *nodes;  
xmlnode *elem, *old, *new;  
...  
if ((nodes = getChildNodes(elem)) &&  
    (old = getNamedItem(nodes, "B", NULL)) &&  
    (new = createElement(ctx, "NEW")))  
{  
    <Thing><A/><B/><C/></Thing>  
    replaceChild(ctx, new, old);  
    <Thing><A/><NEW/><C/></Thing>  
}
```

## replaceData

### Purpose

Replaces the substring at the given character offset and length with a replacement string.



## Syntax

```
void replaceData(xmlctx *ctx, xmlnode *node, ub4 offset,  
                ub4 count, oratext *arg)
```

## Parameters

ctx	(IN)	XML context
node	(IN)	pointer to node
offset	(IN)	start of substring to replace (0 is first character)
count	(IN)	length of old substring
arg	(IN)	replacement text

## Example

```
xmlnode *node;  
...  
getNodeValue(node) -> "every dog has his day"  
replaceData(ctx, node, 6, 3, "man");  
getNodeValue(node) -> "every man has his day"
```

## setAttribute

### Purpose

Create a new attribute for an element. If the named attribute already exists, its value is simply replaced.

### Syntax

```
xmlnode *setAttribute(xmlctx *ctx, xmlnode *elem,  
                      const oratext *name, const oratext *value)
```

### Parameters

ctx	(IN)	XML context
elem	(IN)	pointer to element node
name	(IN)	name of new attribute

value (IN) value of new attribute

## Example

```
xmlnode *elem;
...
<Thing/>
setAttribute(ctx, elem, "attr", "value");
<Thing attr="value"/>
```

## setAttributeNode

### Purpose

Adds a new attribute to the given element. If the named attribute already exists, it is replaced and the user's old pointer (if provided) is set to the old attr. If the attribute is new, it is added and the old pointer is set to NULL. Returns a truth value indicating success.

### Syntax

```
boolean setAttributeNode(xmlctx *ctx, xmlnode *elem,
                        xmlnode *newNode, xmlnode **oldNode)
```

### Parameters

ctx	(IN)	XML context
elem	(IN)	pointer to element node
newNode	(IN)	pointer to new attribute
oldNode	(OUT)	return pointer for old attribute

## Example

```
xmlnode *elem, *attr;
...
if (attr = createAttribute(ctx, "attr", "value"))
{
    <Thing/>
    setAttributeNode(ctx, elem, attr, NULL);
    <Thing attr="value"/>
}
```

```
}

```

## setNamedItem

### Purpose

Sets a new child node in a parent node's map; if an old node exists with same name, replaces the old node (and sets user's pointer, if provided, to it); if no such named node exists, appends node to map and sets pointer to `NULL`.

### Syntax

```
boolean setNamedItem(xmlctx *ctx, xmlnode *parent, xmlnode *node, xmlnode **old)
```

### Parameters

<code>node</code>	(IN)	pointer to node
<code>parent</code>	(IN)	parent to add node to
<code>node</code>	(IN)	new node to add
<code>old</code>	(IN)	pointer to replaced node

### Example

```
xmlnode *elem, *new;
...
if ((new = createElement(ctx, "B")) &&
    setAttribute(ctx, new, "attr", "value"))
{
    <Thing><A/><B/><C/></Thing>
    setNamedItem(ctx, elem, new, NULL);
    <Thing><A/><B attr="value" /><C/></Thing>
}

```

## setNodeValue

### Purpose

Sets the *value* (character data) associated with a node.

## Syntax

```
boolean setNodeValue(xmlnode *node, const oratext *data)
```

## Parameters

node (IN) pointer to node  
data (IN) new data for node

## Example

```
xmlnode *node;  
...  
getNodeValue(node) -> "umbrella"  
setNodeValue(node, "broolly");  
getNodeValue(node) -> "broolly"
```

## setPIData

### Purpose

Sets a Processing Instruction's (PI) data (equivalent to `setNodeValue`). It is not permitted to set the data to `NULL`. Under the DOM spec, this is a method named `setData`.

### Syntax

```
void setPIData(xmlnode *pi, const oratext *data)
```

### Parameters

pi (IN) pointer to PI node  
data (IN) new data for PI

### Example

```
xmlnode *pi;  
...  
<?SKRINKLIT Monster Grendel's tastes are plainish?>  
setPIData(pi, "Breakfast? Just a couple Danish.");
```

```
<?SKRINKLIT Breakfast? Just a couple Danish.??>
```

## splitText

### Purpose

Breaks a TEXT node into two TEXT nodes at the specified offset, keeping both in the tree as siblings. The original node then only contains all the content up to the offset point. And a new node, which is inserted as the next sibling of the original, contains all the old content starting at the offset point.

### Syntax

```
xmlnode *splitText(xmlctx *ctx, xmlnode *old, uword offset)
```

### Parameters

ctx	(IN)	XML context
old	(IN)	original node to split
offset	(IN)	offset of split point

### Example

```
xmlnode *node;  
...  
<FOO>"one of these days"</FOO>  
splitText(ctx, node, 7);  
<FOO>"one of " "these days"</FOO>
```

## substringData

### Purpose

Returns a substring of a node's character data.

### Syntax

```
const oratext *substringData(xmlctx *ctx, const xmlnode *node,  
                             ub4 offset, ub4 count)
```

## Parameters

ctx	(IN)	XML context
node	(IN)	pointer to node
offset	(IN)	offset of start of substring
count	(IN)	length of substring

## Example

```
xmlnode *node;
...
<FOO>"one of these days"</FOO>
substringData(ctx, node, 0, 3) -> "one"
```

## Namespace APIs

Namespace APIs provide an interface that is an extension to the DOM and give information relating to the document namespaces.

XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references. A single XML document may contain elements and attributes (here referred to as a "markup vocabulary") that are defined for and used by multiple software modules. One motivation for this is modularity; if such a markup vocabulary exists which is well-understood and for which there is useful software available, it is better to re-use this markup rather than re-invent it.

Such documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the tags and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element type or attribute name.

These considerations require that document constructs should have universal names, whose scope extends beyond their containing document. This C implementation of XML namespaces provides a mechanism to accomplish this.

Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a namespace prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace. The combination

of the universally managed URI namespace and the document's own namespace produces identifiers that are universally unique. Mechanisms are provided for prefix scoping and defaulting.

URI references can contain characters not allowed in names, so cannot be used directly as namespace prefixes. Therefore, the namespace prefix serves as a proxy for a URI reference. An attribute-based syntax described in the W3C Namespace specification is used to declare the association of the namespace prefix with a URI reference.

The implementation of this C Namespace interface followed the XML Namespace standard of revision REC-xml-names-19990114.

---

## Data Structures and Types

- Oratext
- Xmlattr
- XmlNode



## Functions

`getAttrLocal(xmlattr *attrs)`

**Returns attribute local name.**

`getAttrNamespace(xmlattr *attr)`

**Returns attribute namespace (URI).**

`getAttrPrefix(xmlattr *attr)`

**Returns attribute prefix.**

`getAttrQualified_name(xmlattr *attr)`

**Returns attribute fully qualified name.**

`getNodeLocal(xmlnode *node)`

**Returns node local name.**

`getNodeNamespace(xmlnode *node)`

**Returns node namespace (URI).**

`getNodePrefix(xmlnode *node)`

**Returns node prefix.**

`getNodeQualified_name(xmlnode *node)`

**Returns node qualified name.**

---

## Data Structure and Type Description

```
ORATEXT  
Typedef unsigned char oratext;
```

```
XMLATTR  
  
Typedef struct xmlattr xmlattr;
```

---

---

**Note:** the contents of `xmlattr` are private and must not be accessed by users.

---

---

### XMLNODE

```
Typedef struct xmlnode xmlnode;
```

---

---

**Note:** the contents of `xmlnode` are private and must not be accessed by users.

---

---

## Function Prototypes

### getAttrLocal

**Purpose**

This function returns the local name of this attribute.

**Syntax**

```
const oratext *getAttrLocal(const xmlattr *attr);
```

**Parameters**

attr (IN) - pointer to opaque attribute structure (see getAttribute)

**Comments**

### getAttrNamespace

**Purpose**

This function returns namespace for this attribute.

**Syntax**

```
const oratext *getAttrNamespace(const xmlattr *attr);
```

**Parameters**

attr (IN) - pointer to opaque attribute structure (see getAttribute)

**Comments**

### getAttrPrefix

**Purpose**

This function returns prefix for this attribute.

### Syntax

```
const oratext *getAttrPrefix(const xmlattr *attr);
```

### Parameters

attr (IN) - pointer to opaque attribute structure (see `getAttribute`)

### Comments

## getAttrQualifiedName

### Purpose

This function returns fully qualified name for the attribute.

### Syntax

```
const oratext *getAttrQualifiedName(const xmlattr *attr);
```

### Parameters

attr (IN) - pointer to opaque attribute structure (see `getAttribute`)

### Comments

## getNodeLocal

### Purpose

This function returns the local name of this node.

### Syntax

```
const oratext *getNodeLocal(const xmlnode *node);
```

### Parameters

node (IN) - node to get local name from

## Comments

### getNodeNamespace

#### Purpose

This function returns namespace for this node.

#### Syntax

```
const oratext *getNodeNamespace(const xmlnode *node);
```

#### Parameters

node (IN) - node to get namespace from

## Comments

### getNodePrefix

#### Purpose

This function returns prefix for this node.

#### Syntax

```
const oratext *getNodePrefix(const xmlnode *node);
```

#### Parameters

node (IN) - node to get prefix from

## Comments

### getNodeQualifiedName

#### Purpose

This function returns fully qualified name for this node.

#### Syntax

```
const oratext *getNodeQualifiedName(const xmlnode *node);
```

**Parameters**

node (IN) - node to get name from

**Comments**

## Datatypes

<code>oratext*/String</code>	String pointer (C/C++)
<code>xmlctx</code>	Master XML context
<code>xmlmemcb</code>	Memory callback structure (optional)
<code>xmlsaxcb</code>	SAX callback structure (SAX only)
<code>ub4</code>	32-bit (or larger) unsigned integer
<code>uword</code>	Native unsigned integer
<code>boolean</code>	Boolean value, TRUE or FALSE
<code>oratext</code>	String pointer
<code>xmlcpmod</code>	Content model node modifier
<code>xmlctx</code>	Master XML parser context
<code>xmlnode</code>	Document node
<code>xmlnodes</code>	Array of nodes
<code>xmlntype</code>	Node type enumeration

### **oratext/String**

The basic character pointer type (for C/C++):

```
typedef unsigned char oratext;  
typedef unsigned char String;
```

### **xmlctx**

The top-level XML context:

```
typedef struct xmlctx xmlctx;
```

---

---

**Note:** The contents of `xmlctx` are private and must not be accessed by users.

---

---

### **xmlmemcb**

The memory callback structure passed to `xmlinit`:

```
struct xmlmemcb
{
    void *(*alloc)(void *ctx, size_t size);
    void (*free)(void *ctx, void *ptr);
    void *(*realloc)(void *ctx, void *ptr, size_t size);
};
typedef struct xmlmemcb xmlmemcb
```

## **xmlsaxcb**

The SAX callback structure passed to `xmlInit`:

```
struct xmlsaxcb
{
    sword (*startDocument)(void *ctx);
    sword (*endDocument)(void *ctx);
    sword (*startElement)(void *ctx, const oratext *name,
                          const struct xmlattrs *attrs);
    sword (*endElement)(void *ctx, const oratext *name);
    sword (*characters)(void *ctx, const oratext *ch, size_t len);
    sword (*ignorableWhitespace)(void *ctx, const oratext *ch,
                                 size_t len);
    sword (*processingInstruction)(void *ctx, const oratext *target,
                                  const oratext *data);
    sword (*notationDecl)(void *ctx, const oratext *name,
                          const oratext *publicId,
                          const oratext *systemId);
    sword (*unparsedEntityDecl)(void *ctx, const oratext *name,
                                const oratext *publicId,
                                const oratext *systemId,
                                const oratext *notationName);
    sword (*nsStartElement)(void *ctx, const oratext *qname,
                            const oratext *local,
                            const oratext *namespace,
                            const struct xmlattrs *attrs);
};
typedef struct xmlsaxcb xmlsaxcb;
```

## **ub4**

Unsigned integer with a minimum of four bytes:

```
typedef unsigned int ub4;
```



**uword**

Unsigned integer in the native word size:

```
typedef unsigned int uword;
```

**boolean**

```
typedef int boolean;
```

**oratext**

```
typedef unsigned char oratext;
```

**xmlcpmod**

Content model node modifiers, see `getModifier`.

```
XMLCPMOD_NONE = 0           /* no modifier */  
XMLCPMOD_OPT  = 1           /* '?' optional */  
XMLCPMOD_OMORE = 2           /* '*' zero or more */  
XMLCPMOD_1MORE = 3          /* '+' one or more */
```

## xmlctx

```
typedef struct xmlctx xmlctx;
```

---

---

**Note:** The contents of `xmlctx` are private and must not be accessed by users.

---

---

## xmlnode

```
typedef struct xmlnode xmlnode;
```

---

---

**Note:** The contents of `xmlnode` are private and must not be accessed by users.

---

---

## xmlnodes

```
typedef struct xmlnodes xmlnodes;
```

---

---

**Note:** The contents of `xmlnodes` are private and must not be accessed by users.

---

---

## xmlntype

Parse tree node types, see `getNodeType`. Names and values match DOM specification.

<code>ELEMENT_NODE</code>	= 1	<code>/* element */</code>
<code>ATTRIBUTE_NODE</code>	= 2	<code>/* attribute */</code>
<code>TEXT_NODE</code>	= 3	<code>/* char data not escaped by CDATA */</code>
<code>CDATA_SECTION_NODE</code>	= 4	<code>/* char data escaped by CDATA */</code>
<code>ENTITY_REFERENCE_NODE</code>	= 5	<code>/* entity reference */</code>
<code>ENTITY_NODE</code>	= 6	<code>/* entity */</code>
<code>PROCESSING_INSTRUCTION_NODE</code>	= 7	<code>/* processing instruction */</code>
<code>COMMENT_NODE</code>	= 8	<code>/* comment */</code>
<code>DOCUMENT_NODE</code>	= 9	<code>/* document */</code>
<code>DOCUMENT_TYPE_NODE</code>	= 10	<code>/* DTD */</code>
<code>DOCUMENT_FRAGMENT_NODE</code>	= 11	<code>/* document fragment */</code>
<code>NOTATION_NODE</code>	= 12	<code>/* notation */</code>

---

# XML Schema Processor for C

This chapter describes:

- XML Schema implementation for C

Note: This beta implementation is incomplete at present; not all schema features are implemented. Also, features can be expected to evolve further before stabilizing.

## Schema APIs

The schema API is very simple: initialize, validate,...validate, terminate.

The validation process is go/no-go. Either the document is valid with respect to the schemas or it is invalid. When it is valid, a zero error code is returned. When it is invalid, a non-zero error code is returned indicating the problem. There is no distinction between warnings and errors; all problems are errors and considered fatal: validation stops immediately.

As schemas are encountered, they are loaded and preserved in the schema context. No schema is loaded more than once during a session. There is no clean up call similar to `xmlclean`. Hence, if you need to release all memory and reset state before validating a new document, you must terminate the context and start over.

## Function/Method Index

**Table 8-1**

Function/Method	Description
<code>schemaInitialize</code>	Initialize the schema processor
<code>schemaValidate</code>	Validate an instance document against a schema
<code>schemaTerminate</code>	Terminates (tears down) the schema processor

## Functions

### `schemaInitialize`

#### **Purpose**

Initializes the XML schema processor. Must be called before the processor can be used to validate any documents.

```
xsdctx *schemaInitialize(xmlctx *ctx, uword *err)
```

**Parameters**

ctx (IN) XML parser context  
err (OUT) Returned error code

**Comments**

The XML parser context is used to allocate memory for the schema context.

Note this is not the context of the instance document to be validated.

The schema context is returned, and must be passed to all subsequent schema functions. This context pointer is opaque-- you cannot reference its members. If the return context is NULL, initialization failed and `err` will be set with the numeric error code indicating the problem.

## schemaValidate

### Purpose

Validates an instance document against a schema or schemas.

```
uword schemaValidate(xsdctx *sctx, xmlctx *inst, oratext *schema)
```

### Parameters

`sctx` (IN) Schema context  
`inst` (IN) Instance document context  
`schema` (IN) URL of default schema

### Comments

This is the function which actually performs schema validation. The schema context returned by `schemaInitialize` must be passed in.

The document to be validated is specified by the XML parser context `inst` used to parse the document. Note the document must already have been parsed.

If no schemas are explicitly referenced in the instance document, the default schema (specified by URL) is assumed. If the document *does* specify all necessary schemas, and a default schema is also supplied, the default will be ignored. If the document does not reference any schemas and no default is supplied, an error will result.

## schemaTerminate

### Purpose

Terminates (shuts down) the schema processor, freeing all memory allocated on the original XML parser context passed to `schemaInitialize`.

```
void schemaTerminate(xsdctx *sctx)
```

### Parameters

`sctx` (IN) Schema context

**Comments**

After termination, the schema context is no longer valid. To continue using the schema processor, a new schema must be created with `schemaInitialize`.





# Part IV

---

---

## XDK for C++ Packages

This section contains the following chapters:

- [Chapter 9, "XML Parser for C++"](#)
- [Chapter 10, "Oracle XML Class Generator \(C++\)"](#)
- [Chapter 11, "XML Schema Processor for C++"](#)



---

# XML Parser for C++

This chapter describes the following sections:

- Class APIs
- Parser APIs
- C++ Sax APIs
- C++ DOM APIs

---

## Class: Attr

This class contains methods for accessing the name and value of a single document node attribute.

getName	Return name of attribute
getValue	Return "value" (definition) of attribute
getSpecified	Return attribute's "specified" flag value
setValue	Set an attribute's value

### getName

#### Function

Return name of attribute

#### Prototype

```
String getName()
```

#### Arguments

None

#### Returns

*String* -- Name of attribute

### getValue

#### Function

Return "value" (definition) of attribute

#### Prototype

```
String getValue()
```

**Arguments**

None

**Returns**

Value of attribute

**getSpecified****Function**

Return value of attribute's "specified" flag. The DOM says:

If this attribute was explicitly given a value in the original document, this is true; otherwise, it is false. Note that the implementation is in charge of this attribute, not the user. If the user changes the value of the attribute (even if it ends up having the same value as the default value) then the specified flag is automatically flipped to true. To re-specify the attribute as the default value from the DTD, the user must delete the attribute. The implementation will then make a new attribute available with specified set to false and the default value (if one exists).

**Prototype**

```
boolean getSpecified()
```

**Arguments**

None

**Returns**

Value of specified flag

**setValue****Function**

Sets an attribute's "value"

**Prototype**

```
void setValue(String value)
```

**Arguments**

```
value -- Attribute's new value
```

**Returns**

Value of attribute

---

## Class: CDATASection

This class implements the CDATA node type, a subclass of **Text**.

---

## Class: Comment

This class implements the COMMENT node type, a subclass of **CharacterData**.



## Class: Document

This class contains methods for creating and retrieving nodes.

<code>createAttribute</code>	Create an <code>ATTRIBUTE</code> node
<code>createCDATASection</code>	Create a <code>CDATA</code> node
<code>createComment</code>	Create a <code>COMMENT</code> node
<code>createDocumentFragment</code>	Create a <code>DOCUMENT_FRAGMENT</code> node
<code>createElement</code>	Create an <code>ELEMENT</code> node
<code>createEntityReference</code>	Create an <code>ENTITY_REFERENCE</code> node
<code>createProcessingInstruction</code>	Create a <code>PROCESSING_INSTRUCTION</code> node
<code>createTextNode</code>	Create a <code>TEXT</code> node
<code>getElementsByTagName</code>	Select nodes based on tag name
<code>getImplementation</code>	Return DTD for document

### `createAttribute`

#### **Function**

Create a new attribute node. Use `setValue` to set its value.

#### **Prototype**

`Attr*` `createAttribute`(String name)

#### **Arguments**

name -- name of attribute

#### **Returns**

`Attr*` -- pointer to created node

## createCDATASection

### Function

Create a new CDATA node with the given contents.

### Prototype

```
Attr* createCDATASection(String name)
```

### Arguments

data -- contents of node

### Returns

CDATASection\* -- pointer to created node

## createComment

### Function

Create a new comment node with the given contents.

### Prototype

```
Comment* createComment(String data)
```

### Arguments

data -- contents of node

### Returns

Comment\* -- pointer to created node

## createDocumentFragment

### Function

Create a new document fragment node.

### Prototype

```
DocumentFragment* createDocumentFragment()
```

**Arguments**

None

**Returns**

DocumentFragment\* -- pointer to created node

**createElement****Function**

Create a new element node with the given (tag) name.

**Prototype**

Element\* createElement(String tagName)

**Arguments**

tagName -- element's tagname

**Returns**

Element\* -- pointer to created node

**createEntityReference****Function**

Create a new entity reference node.

**Prototype**

EntityReference\* createEntityReference(String name)

**Arguments**

name -- name of entity to reference

**Returns**

EntityReference\* -- pointer to created node

## createProcessingInstruction

### Function

Create a new processing instruction node.

### Prototype

```
ProcessingInstruction* createProcessingInstruction(String target, String data)
```

### Arguments

target -- target part of PI

data -- data for node

### Returns

ProcessingInstruction\* -- pointer to created node

## createTextNode

### Function

Create a new TEXT node.

### Prototype

```
Text* createTextNode(String data)
```

### Arguments

data -- data for node

### Returns

Text\* -- pointer to created node

## getElementsByTagName

### Function

Returns a NodeList of all the Elements with a given tag name in the order in which they would be encountered in a preorder traversal of the Document tree. The special value "\*" matches all tags.

**Prototype**

```
NodeList* getElementsByTagName(String tagname)
```

**Arguments**

tagname -- tag name to select

**Returns**

NodeList\* -- list of matches, NULL if none

**getImplementation****Function**

Returns the DOMImplementation structure, currently useless. Perhaps it will be used in later DOM versions.

**Prototype**

```
DOMImplementation* getImplementation()
```

**Arguments**

None

**Returns**

DOMImplementation\* -- pointer to structure

---

## Class: DocumentType

This class contains methods for accessing information about the Document Type Definition (DTD) of a document.

getName	Return name of DTD
getEntities	Return NamedNodeMap of DTD's (general) entities
getNotations	<b>Return NamedNodeMap of DTD's notations</b>

### getName

#### Function

Return name of DTD

#### Prototype

```
String getName()
```

#### Arguments

None

#### Returns

String -- Name of DTD

### getEntities

#### Function

Returns map of DTD's (general) entities

#### Prototype

```
NamedNodeMap* getEntities()
```

#### Arguments

None

### Returns

NamedNodeMap\* -- map of entities

## getNotations

### Function

Return map of DTD's notations

### Prototype

```
NamedNodeMap* getNotations()
```

### Arguments

None

### Returns

NamedNodeMap\* -- map of notations

---

## Class: DOMImplementation

This class contains methods relating to the specific DOM implementation supported by the parser.

`hasFeature`      Detect if the named feature is supported

### hasFeature

#### Function

Test if the DOM implementation implements a specific feature.

#### Prototype

```
boolean hasFeature(DOMString feature, DOMString version)
```

#### Arguments

`feature` -- The package name of the feature to test. In Level 1, the legal values are "HTML" and "XML" (case-insensitive)

`version` -- This is the version number of the package name to test. In Level 1, this is the string "1.0". If the version is not specified, supporting any version of the feature will cause the method to return true.

#### Returns

`boolean` -- feature is supported



## Class: Element

This class contains methods pertaining to element nodes.

<code>getTagName</code>	Return the node's tag name
<code>getAttribute</code>	Select an attribute given its name
<code>setAttribute</code>	Create a new attribute given its name and value
<code>removeAttribute</code>	Remove an attribute given its name
<code>getAttributeNode</code>	Remove an attribute given its name
<code>setAttributeNode</code>	Add a new attribute node
<code>removeAttributeNode</code>	Remove an attribute node
<code>getElementsByTagName</code>	Return a list of element nodes with the given tag name
<code>normalize</code>	"Normalize" an element (merge adjacent text nodes)

### `getTagName`

#### Function

Return the tag name of the element. The DOM says: "...even though there is a generic `nodeName` attribute on the `Node` interface, there is still a `tagName` attribute on the `Element` interface; these two attributes must contain the same value, but the Working Group considers it worthwhile to support both, given the different constituencies the DOM API must satisfy

#### Prototype

```
String getTagName()
```

#### Arguments

None

#### Returns

String -- Tag name of node

## getAttribute

### Function

Return "value" (definition) of named attribute

### Prototype

```
String getAttribute(String name)
```

### Arguments

name -- name of attribute

### Returns

Value of attribute

## setAttribute

### Function

Create a new attribute

### Prototype

```
Attr* setAttribute(String name, String value)
```

### Arguments

name -- name of new attribute

value -- value of new attribute

### Returns

Pointer to create attribute

## removeAttribute

### Function

Removes the named attribute

### Prototype

```
void removeAttribute(String name)
```

**Arguments**

name -- name of attribute to remove

**Returns**

None

**getAttributeNode****Function**

Return pointer to named attribute

**Prototype**

```
Attr* getAttributeNode(DOMString name)
```

**Arguments**

name -- name of attribute

**Returns**

Attr\* -- pointer to attribute, or NULL if none such

**setAttributeNode****Function**

Set (add) new attribute

**Prototype**

```
boolean setAttributeNode(Attr* newAttr, Attr** oldAttr)
```

**Arguments**

newAttr -- pointer to new attribute

oldAttr -- returned pointer to replaced attribute

**Returns**

boolean -- success

## removeAttributeNode

### Function

Remove the named attribute

### Prototype

```
Attr* removeAttributeNode(Attr* oldAttr)
```

### Arguments

oldAttr -- attribute to remove

### Returns

Attr\* -- oldAttr passed back

## getElementsByTagName

### Function

Create a list of matching elements

### Prototype

```
NodeList* getElementsByTagName(DOMString name)
```

### Arguments

name -- tagname to match, "\*" for all

### Returns

NodeList\* -- list of matches

## normalize

### Function

Normalize an element, i.e. merge all adjacent TEXT nodes

### Prototype

```
void normalize(void)
```

**Arguments**

None

**Returns**

None

---

## Class: Entity

This class implements the ENTITY node type, a subclass of **Node**.

`getNotationName` Return entity's NDATA (notation name)

`getPublicId` Return entity's public ID

`getSystemId` **Return entity's system ID**

### `getNotationName`

#### **Function**

Return an entity node's notation name (NDATA)

#### **Prototype**

```
String* getNotationName()
```

#### **Arguments**

None

#### **Returns**

String -- node's NDATA

### `getPublicId`

#### **Function**

Return an entity node's public ID

#### **Prototype**

```
String getPublicId()
```

#### **Arguments**

None

**Returns**

String -- entity's public ID

**getSystemId**

**Function**

Return an entity node's system ID

**Prototype**

String getSystemId()

**Arguments**

None

**Returns**

String -- entity's system ID

---

## Class: EntityReference

This class implements the ENTITY\_REFERENCE node type, a subclass of **Node**.



## Class: NamedNodeMap

This class contains methods for accessing the number of nodes in a node map and fetching individual nodes.

<code>item</code>	Return <i>n</i> th node in map.
<code>getLength</code>	Return number of nodes in map
<code>getNamedItem</code>	Select a node by name
<code>setNamedItem</code>	Set a node into the map
<code>removeNamedItem</code>	Remove the named node from map

### item

#### Function

Return *n*th node in node map

#### Prototype

```
Node* item(size_t index)
```

#### Arguments

`size_t index`-- zero-based node number

#### Returns

`Node*` -- Pointer to `index`'th node in map

### getLength

#### Function

Return number of nodes in map

#### Prototype

```
size_t getLength()
```

### Arguments

None

### Returns

size\_t -- Number of nodes in map

## getNamedItem

### Function

Selects the node with the given name from the map

### Prototype

```
Node* getNamedItem(String name)
```

### Arguments

name -- Name of node to select

### Returns

Node\* -- Pointer to named node, NULL if none such exists

## setNamedItem

### Function

Adds a node to the map, replacing any node that already exists with the same name

### Prototype

```
boolean setNamedItem(Node *node, Node **old)
```

### Arguments

node -- Name of node to add

old -- Pointer to replaced node, NULL if node is new

### Returns

boolean -- Success

## removeNamedItem

### Function

Removes the node with the given name from the node map

### Prototype

```
Node* removeNamedItem(String name)
```

### Arguments

name -- Name of node to remove

### Returns

Node\* -- Pointer to removed node, NULL if none such exists

## Class: Node

---

This class contains methods for details about a document node

<code>appendChild</code>	Append a new child to the end of the current node's list of children
<code>cloneNode</code>	Clone an existing node and optionally all its children
<code>getAttributes</code>	Return structure contains all defined node attributes
<code>getChildNode</code>	Return specific indexed child of given node
<code>getChildNodes</code>	Return structure contains all child nodes of given node
<code>getFirstChild</code>	Return first child of given node
<code>getLastChild</code>	Return last child of given node
<code>getLocal</code>	Returns the local name of the node
<code>getNamespace</code>	Return a node's namespace
<code>getNextSibling</code>	Return a node's next sibling
<code>getName</code>	Return name of node
<code>getType</code>	Return numeric type-code of node
<code>getValue</code>	Return "value" (data) of node
<code>getOwnerDocument</code>	Return document node which contains a node
<code>getParentNode</code>	Return parent node of given node
<code>getPrefix</code>	Returns the namespace prefix for the node
<code>getPreviousSibling</code>	Returns the previous sibling of the current node
<code>getQualifiedName</code>	Return namespace qualified node of given node
<code>hasAttributes</code>	Determine if node has any defined attributes
<code>hasChildNodes</code>	Determine if node has children
<code>insertBefore</code>	Insert new child node into a node's list of children
<code>numChildNodes</code>	Return count of number of child nodes of given node
<code>removeChild</code>	Remove a node from the current node's list of children

replaceChild	Replace a child node with another
setValue	Sets a node's value (data)

## appendChild

### Function

Append a new child to the current node's list of children

### Prototype

```
Node* appendChild(Node *newChild)
```

### Arguments

newChild -- new child node

### Returns

Node\* -- newChild is passed back

## cloneNode

### Function

Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes. The duplicate node has no parent (parentNode returns NULL).

Cloning an Element copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes, but this method does not copy any text it contains unless it is a deep clone, since the text is contained in a child Text node. Cloning any other type of node simply returns a copy of this node.

### Prototype

```
Node* cloneNode(boolean deep)
```

### Arguments

deep -- recursion flag

### Returns

Node\* -- Pointer to new clone

## getAttributes

### Function

Return structure of all attributes for node

### Prototype

```
NamedNodeMap* getAttributes()
```

### Arguments

None

### Returns

NamedNodeMap\* -- Pointer to structure describing all attributes for node, or NULL if no attributes are defined

## getChildNode

### Function

Return one of the node's children

### Prototype

```
Node* getChildNode(ushort index)
```

### Arguments

`index` -- child number, starting at 0

### Returns

Node\* -- Pointer to `index`'th child of node

## getChildNodes

### Function

Return node's children

### Prototype

```
NodeList* getChildNodes()
```

**Arguments**

None

**Returns**

NodeList\* -- Pointer to structure describing all the node's children

**getFirstChild****Function**

Return the node's first child

**Prototype**

Node\* getFirstChild()

**Arguments**

None

**Returns**

Node\* -- Pointer to the node's first child

**getLastChild****Function**

Return the node's last child

**Prototype**

Node\* getLastChild()

**Arguments**

None

**Returns**

Node\* -- Pointer to the node's last child

## getLocal

### Function

Return the node's local name

### Prototype

```
String getLocal()
```

### Arguments

None

### Returns

String -- node's local name

## getNamespace

### Function

Return the node's namespace

### Prototype

```
String getNamespace()
```

### Arguments

None

### Returns

String -- node's namespace (may be NULL)

## getNextSibling

### Function

Returns the next sibling of the node, that is, the child of its parent which comes next

### Prototype

```
Node* getNextSibling()
```



**Arguments**

None

**Returns**

Node\* -- Node's next sibling, NULL if last child

**getName****Function**

Return name of node, or NULL if the node has no name

**Prototype**`String getName()`**Arguments**

None

**Returns**`String` -- Node's name**getType****Function**

Return numeric type-code for node

**Prototype**`short getType()`**Arguments**

None

**Returns**`short` -- Node's numeric type code**Type codes:**`ELEMENT_NODE`

ATTRIBUTE\_NODE  
TEXT\_NODE  
CDATA\_SECTION\_NODE  
ENTITY\_REFERENCE\_NODE  
ENTITY\_NODE  
PROCESSING\_INSTRUCTION\_NODE  
COMMENT\_NODE  
DOCUMENT\_NODE  
DOCUMENT\_TYPE\_NODE  
DOCUMENT\_FRAGMENT\_NODE  
NOTATION\_NODE

## getValue

### Function

Return "value" (data) of node, or NULL if the node has no value

### Prototype

```
String getValue()
```

### Arguments

None

### Returns

String -- Node's "value"

## getOwnerDocument

### Function

Return document node which contains the current node

### Prototype

```
Document* getOwnerDocument()
```

### Arguments

None

**Returns**

Document\* -- Pointer to document node

**getParentNode****Function**

Return node's parent

**Prototype**

Node\* getParentNode()

**Arguments**

None

**Returns**

Node\* -- Pointer to the node's parent node

**getPrefix****Function**

Return the namespace prefix of node

**Prototype**

String getPrefix()

**Arguments**

None

**Returns**

String -- Node's namespace prefix, may be NULL

**getPreviousSibling****Function**

Returns the previous sibling of the node, that is, the child of its parent which came before

### Prototype

Node\* getPreviousSibling()

### Arguments

None

### Returns

Node\* -- Node's previous sibling, NULL if first child

## getQualifiedName

### Function

Return the fully qualified (namespace) name of node

### Prototype

String getQualifiedName()

### Arguments

None

### Returns

String -- Node's qualified name

## hasAttributes

### Function

Determine if node has any defined attributes

### Prototype

boolean hasAttributes()

### Arguments

None

### Returns

boolean -- TRUE if node has attributes

## hasChildNodes

### Function

Determine if node has any children

### Prototype

```
boolean hasChildNodes()
```

### Arguments

None

### Returns

boolean -- TRUE if node has child nodes

## insertBefore

### Function

Insert a new child node into the list of children of a parent, before the reference node. If refChild is NULL, appends the new node to the end.

### Prototype

```
Node* insertBefore(Node *newChild, Node *refChild)
```

### Arguments

newChild -- new node to insert

refChild -- reference node; new node comes before

### Returns

Node\* -- newChild passed back

## numChildNodes

### Function

Return count of node's children

### Prototype

`uword numChildNodes()`

### Arguments

None

### Returns

`uword` -- Number of children this node has (might be 0)

## removeChild

### Function

Remove a child node from the current node's list of children

### Prototype

`Node* removeChild(Node *oldChild)`

### Arguments

`oldChild` -- old node being removed

### Returns

`Node*` -- `oldChild` is passed back

## replaceChild

### Function

Replace one node with another. `newChild` replaces `oldChild` in the list of children in `oldChild`'s parent.

### Prototype

`Node* replaceChild(Node *newChild, Node *oldChild)`

### Arguments

`newChild` -- new replacement node

`oldChild` -- old node being replaced

**Returns**

Node\* -- oldChild is passed back

**setValue****Function**

Sets a node's "value" (data)

**Prototype**

```
void setValue(String data)
```

**Arguments**

data -- New data for node

**Returns**

void

---

## Class: NodeList

This class contains methods for extracting nodes from a NodeList

<code>item</code>	Return <i>n</i> th node in list
<code>getLength</code>	Return number of nodes in list

### item

#### Function

Return *n*th node in node list

#### Prototype

```
Node* item(size_t index)
```

#### Arguments

`size_t index`-- zero-based node number

#### Returns

Pointer to `index`'th node in list

### getLength

#### Function

Return number of nodes in list

#### Prototype

```
size_t getLength()
```

#### Arguments

None

#### Returns

Number of nodes in list



## Class: Notation

This class implements the NOTATION node type, a subclass of **Node**.

<code>getData</code>	Return notation's data
<code>getTarget</code>	Return notation's target
<code>setData</code>	Set notation's data

### `getData`

#### **Function**

Return a notation's data

#### **Prototype**

```
String getData()
```

#### **Arguments**

None

#### **Returns**

String -- node's data

### `getTarget`

#### **Function**

Return a notation's target

#### **Prototype**

```
String getTarget()
```

#### **Arguments**

None

**Returns**

String -- node's target

**setData**

**Function**

Set a notation's data

**Prototype**

void setData(String data)

**Arguments**

data -- new data

**Returns**

None

## Class: ProcessingInstruction

This class implements the PROCESSING\_INSTRUCTION node type, a subclass of **Node**.

getData	Return the PI's data
getTarget	Return the PI's target
setData	Set the PI's data

### getData

#### Function

Return data for a processing instruction

#### Prototype

```
String getData()
```

#### Arguments

None

#### Returns

String -- PI's data

### getTarget

#### Function

Return a processing instruction's target value

#### Prototype

```
String getTarget()
```

#### Arguments

None

**Returns**

String -- target value

**setData**

**Function**

Set the data for a processing instruction

**Prototype**

void setData(String data)

**Arguments**

data -- PI's new data

**Returns**

None

## Class: Text

This class contains methods for accessing and modifying the data associated with text nodes (subclasses CharacterData).

`splitText`                      Get data (value) of text node

### `splitText`

#### **Function**

Split a text node in two. The original node retains its data up to the split point, and the remaining data is turned into a new text node which follows.

#### **Prototype**

```
Text* splitText(unsigned long offset)
```

#### **Arguments**

`offset` -- split point

#### **Returns**

`Text*` -- Pointer to new text node

## Class: XMLParser

This class contains top-level methods for invoking the parser and returning high-level information about a document.

<code>xmlinit</code>	Initialize XML parser
<code>xmlterm</code>	Terminate XML parser
<code>xmlparse</code>	Parse a document from a file
<code>xmlParseBuffer</code>	Parse a document from a buffer
<code>getContent</code>	Returns the content model for an element
<code>getModifier</code>	Returns the modifier ('?', '*' or '+') for a content-model node
<code>getDocument</code>	Returns the root node of a parsed document
<code>getDocumentElement</code>	Returns the root element (node) of a parsed document
<code>getDocType</code>	Returns the document type string
<code>isStandAlone</code>	Returns the value of the standalone flag
<code>isSingleChar</code>	Determine if document encoding is single or multibyte.
<code>getEncoding</code>	Return name of document's character encoding.

### `xmlinit`

#### Function

Initialize XML parser

#### Prototype

```
uword xmlinit(oratext *encoding,  
              void (*msghdlr)(void *msgctx, oratext *msg, ub4 errcode),  
              void *msgctx, lpxsaxcb *saxcb, void *saxcbctx, oratext *lang)
```

**Arguments**

`encoding` -- Input file's encoding, default **UTF8**  
`msghdlr` -- Error message callback  
`msgctx` -- User-defined context pointer passed to `msghdlr`  
`saxcb` -- SAX callback structure (iff using SAX)  
`saxcbctx` -- User-defined SAX context structure passed to SAX callback functions  
`lang` -- Language for error message (not used)

**Returns**

`uword` -- Numeric error code, 0 meaning success

**xmlterm****Function**

Terminate XML parser, tear down, free memory, etc

**Prototype**

```
void xmlterm()
```

**Arguments**

None

**Returns**

void

**xmlparse****Function**

Parses a document

**Prototype**

```
uword xmlparse(oratext *doc, oratext *encoding, ub4 flags)
```

**Arguments**

`doc` -- document path  
`encoding` -- document's encoding

flags -- Mask of flag bits

**Flags:**

XML\_FLAG\_VALIDATE -- Validate document against DTD

XML\_FLAG\_DISCARD\_WHITESPACE -- Discard ignorable whitespace

**Returns**

uword -- Error code, 0 on success

## xmlparseBuffer

**Function**

Parses a document

**Prototype**

uword xmlparseBuffer(oratext \*buffer, size\_t len, oratext \*encoding, ub4 flags)

**Arguments**

buffer -- buffer containing document to parse

len -- length of document

encoding -- document's encoding

flags -- Mask of flag bits

**Flags:**

XML\_FLAG\_VALIDATE -- Validate document against DTD

XML\_FLAG\_DISCARD\_WHITESPACE -- Discard ignorable whitespace

**Returns**

uword -- Error code, 0 on success

## getContent

**Function**

Returns the content model for a node. Content model nodes are Nodes and can be traversed and examined with the same functions as the parsed document.



### Prototype

Node\* getContent(Node \*node)

### Arguments

node -- node whose content model to return

### Returns

Node\* -- root node of content model tree

## getModifier

### Function

Returns the modifier for a content model node. The modifier is one of XMLCPMOD\_NONE (no modifier), XMLCPMOD\_OPT ('?', optional), XMLCPMOD\_0MORE (\*, zero or more), or XMLCPMOD\_1MORE ('+', one or more).

### Prototype

xmlcpmod getContent(Node \*node)

### Arguments

node -- content model node whose modifier to return

### Returns

xmlcpmod -- enumeration as described above

## getDocument

### Function

After a document has been successfully parsed, returns a pointer to the root node of the document. Compare with `getDocumentElement` which returns the root *element* node.

### Prototype

Node\* getDocument()

### Arguments

None

### Returns

Node\* -- Pointer to root node of document

## getDocumentElement

### Function

After a document has been successfully parsed, returns a pointer to the root element (node) of the document

### Prototype

```
Element* getDocumentElement()
```

### Arguments

None

### Returns

Element\* -- Pointer to root element (node) of document

## getDocType

### Function

Returns a pointer to a "DocType" structure which describes the DTD

### Prototype

```
DocumentType* getDocType()
```

### Arguments

None

### Returns

DocumentType\* -- Pointer to DTD descriptor

## isStandalone

### Function

Returns TRUE if the document is specified as standalone on the `<?xml?>` line, FALSE otherwise

### Prototype

```
boolean isStandalone()
```

### Arguments

None

### Returns

`boolean` -- Value of standalone flag

## isSingleChar

### Purpose

Returns a flag which specifies whether the current document is encoded as single-byte characters (i.e. ASCII), or multi-byte characters (e.g. UTF-8).

### Syntax

```
boolean isSingleChar()
```

### Parameters

None

### Comments

Compare to `getEncoding`, which returns the actual name of the document's encoding.

## getEncoding

### Purpose

Returns the name of the current document's character encoding scheme (e.g., "ASCII", "UTF8", etc).

## Syntax

String getEncoding()

## Parameters

None

## Comments

Compare to `isSingleChar` which just returns a boolean flag saying whether the current encoding is single or multi-byte.

## C++ SAX API

The SAX API is based on callbacks. Instead of the entire document being parsed and turned into a data structure which may be referenced (by the DOM interface), the SAX interface is serial. As the document is processed, appropriate SAX user callback functions are invoked. Each callback function returns an error code, zero meaning success, any non-zero value meaning failure. If a non-zero code is returned, document processing is stopped.

To use SAX, an `xmlsaxcb` structure is initialized with function pointers and passed to the `xmlinit()` call. A pointer to a user-defined context structure may also be included; that context pointer will be passed to each SAX function.

Note this SAX functionality is identical to the C version.

## SAX callback structure

```
typedef struct
{
    sword (*)(void *ctx);
    sword (*)(void *ctx);
    sword (*)(void *ctx, const oratext *name, struct xmlarray *attrs);
    sword (*)(void *ctx, const oratext *name);
    sword (*)(void *ctx, const oratext *ch, size_t len);
    sword (*)(void *ctx, const oratext *ch, size_t len);
    sword (*)(void *ctx, const oratext *target, const oratext *data);
    sword (*)(void *ctx, const oratext *name,
              const oratext *publicId, const oratext *systemId);
    sword (*)(void *ctx, const oratext *name, const oratext *publicId,
              const oratext *systemId, const oratext
*notationName);
```

```
    sword (*)(void *ctx, const oratext *qname,  
              const oratext *local, const oratext *namespace);  
} xmlsaxcb;
```

## startDocument

### Function

Called once when document processing is first starting

### Prototype

```
sword startDocument(void *ctx)
```

### Arguments

ctx -- User-defined context as passed to initialize()

### Returns

sword -- Error code, 0 for success, non-0 for error.

## endDocument

### Function

Called once when document processing is finished

### Prototype

```
sword endDocument(void *ctx)
```

### Arguments

ctx -- User-defined context as passed to initialize()

### Returns

sword -- Error code, 0 for success, non-0 for error.

## startElement

### Function

Called once for each new document element

### Prototype

```
sword startElement(void *ctx, const oratext *name, struct xmlarray *attrs)
```

### Arguments

`ctx` -- User-defined context as passed to `initialize()`

`name` -- name of node

`attrs` -- array of node's attributes

### Returns

`sword` -- Error code, 0 for success, non-0 for error.

## endElement

### Function

Called once when each document element closes

### Prototype

```
sword endElement(void *ctx, const oratext *name)
```

### Arguments

`ctx` -- User-defined context as passed to `initialize()`

`name` -- name of node

### Returns

`sword` -- Error code, 0 for success, non-0 for error.

## characters

### Function

Called for each piece of literal text

### Prototype

```
sword characters(void *ctx, const oratext *ch, size_t len)
```

**Arguments**

`ctx` -- User-defined context as passed to `initialize()`

`ch` -- pointer to text

`len` -- number of character in text

**Returns**

`sword` -- Error code, 0 for success, non-0 for error.

**IgnorableWhitespace****Function**

Called for each piece of ignorable (non-significant) whitespace

**Prototype**

```
sword ignorableWhitespace(void *ctx, const oratext *ch, size_t len)
```

**Arguments**

`ctx` -- User-defined context as passed to `initialize()`

`ch` -- pointer to whitespace text

`len` -- number of characters of whitespace

**Returns**

`sword` -- Error code, 0 for success, non-0 for error.

**processingInstruction****Function**

Called once for each PI (Processing Instruction)

**Prototype**

```
sword processingInstruction(void *ctx, const oratext *target,  
                           const oratext *data)
```

**Arguments**

`ctx` -- User-defined context as passed to `initialize()`

`target` -- PI target

`data` -- PI data

**Returns**

`sword` -- Error code, 0 for success, non-0 for error.

**notationDecl****Function**

Called once for each NOTATION

**Prototype**

```
sword notationDecl(void *ctx, const oratext *name,  
                  const oratext *publicId, const oratext *systemId)
```

**Arguments**

`ctx` -- User-defined context as passed to `initialize()`

`name` -- name of notation

`publicId` -- Public ID

`systemId` -- System ID

**Returns**

`sword` -- Error code, 0 for success, non-0 for error.

**unparsedEntityDecl****Function**

Called once for each unparsed entity declaration

**Prototype**

```
sword unparsedEntityDecl(void *ctx, const oratext *name, const oratext  
*publicId,
```



```
const oratext *systemId, const oratext *notationName)
```

### Arguments

`ctx` -- User-defined context as passed to `initialize()`

`name` -- name of entity

`publicId` -- Public ID

`systemId` -- System ID

`notationName` -- notation name

### Returns

`sword` -- Error code, 0 for success, non-0 for error.

## nsStartElement

### Function

Namespace variant of `startElement`: Called once for each new document element, when the element uses an explicit namespace

### Prototype

```
sword startElement(void *ctx, const oratext *qname,  
                  const oratext *local, const oratext *namespace)
```

### Arguments

`ctx` -- User-defined context as passed to `initialize()`

`qname` -- qualified namespace

`local` -- umm

`namespace` -- yes, well

### Returns

`sword` -- Error code, 0 for success, non-0 for error.

## C++ *DOM* API's

### Class (Subclass)

Attr	Node
CDATASection	Text
CharacterData	Node
Comment	CharacterData
Document	Node
DocumentFragment	Node
DocumentType	Node
DOMImplementation	
Element	Node
Entity	Node
EntityReference	Node
NamedNodeMap	
Node	
NodeList	
Notation	Node
ProcessingInstruction	Node
Text	CharacterData

---

## Oracle XML Class Generator (C++)

This chapter details the XML Class Generator for C++ and how it defines each element. It contains:

- Document Type Defintiions
- XML Documents with DTD
- C++ Source files
- Sample/Example Usage
- Class: XMLClass Generator
- Class: generated

## Overview of the XML Class Generator for C++

The XML Class Generator takes a Document Type Definition (DTD) and generates classes for each defined element. Those classes are then used in a C++ program to construct XML documents conforming to the DTD. Supported operating systems are Solaris 2.6, Linux 2.2, and NT 4 (Service Pack 3 and above).

### Input

Input is an XML document containing a DTD. The document body itself is ignored; only the DTD is relevant, though the dummy document must conform to the DTD. The underlying XML parser only accepts file names for the document and associated external entities. In future releases, no dummy document will be required, and URIs for additional protocols will be accepted.

The following input file encodings are supported:

UTF-8, UTF-16, US-ASCII, ISO-10646-UCS-2, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, EUC-JP, SHIFT\_JIS, BIG5, GB2312, KOI8-R, EBCDIC-CP-US, EBCDIC-CP-CA, EBCDIC-CP-NL, EBCDIC-CP-WT, EBCDIC-CP-DK, EBCDIC-CP-NO, EBCDIC-CP-FI, EBCDIC-CP-SE, EBCDIC-CP-IT, EBCDIC-CP-ES, EBCDIC-CP-GB, EBCDIC-CP-FR, EBCDIC-CP-HE, EBCDIC-CP-BE, EBCDIC-CP-CH, EBCDIC-CP-ROECE, EBCDIC-CP-YU, and EBCDIC-CP-IS.

In addition, any character set specified in Appendix A, Character Sets, of the Oracle National Language Support Guide may be used. The default encoding is UTF-8. It is recommended that you set the default encoding explicitly if using only single byte character sets (such as US-ASCII or any of the ISO-8859 character sets) for performance up to 25% faster than with multibyte character sets such as UTF-8.

### Output

Output is a pair of C++ source files, .cpp and .h, named after the DTD. Constructors are provided for each class (element) that allow an object to be created in two different ways: initially empty, then adding the children or data after the initial creation, or created with the initial full set of children or initial data. A method is provided for #PCDATA (and Mixed) elements to set the data and, when appropriate, set an element's attributes.

## Relevant XML Standards

The W3C recommendation for Extensible Markup Language (XML) 1.0

The W3C recommendation for Document Object Model Level 1 1.0

The W3C proposed recommendation for Namespaces in XML

The Simple API for XML (SAX) 1.0

## sample/Example usage

### xmlcg Usage

The standalone parser may be called as an executable by invoking

```
bin/xmlcg like  
xmlcg [flags] &lt;XML document&gt;  
where the optional flags are as follows:
```

```
-d  
directory  
Specify output directory (default is current directory)
```

```
-e  
encoding  
Specify default input file encoding
```

```
-h help  
show this usage help
```

**Class: XMLClassGenerator**

This class contain the method for generating classes based on a DTD.

**METHOD INDEX**

generate    Generate classes

**METHODS**

## generate

**Function:**

Generates classes for the given DTD. Two files are created in the output directory `outdir` (or in the current directory if `outdir` is `NULL`): `DTDname.h` and `DTDname.cpp`, both named after the DTD. One class is generated for each defined element in the DTD.

**Prototype:**

```
uword generate(DocumentType *dtd, char *outdir)
```

**Arguments:**

`dtd` -- DTD whose elements to generate classes for

`outdir` -- directory in which to place output files

**Returns:**

`uword` -- error code, 0 on success

---

## Class: generated

A generated class is produced for each element defined in the DTD. It has the same name as the element.

Constructors are provided which create an empty element, or make it with an initial set of children or data. Methods are provided to add children or data after construction, and to set attributes. There are two styles of creation: make an empty element, then add the children one at a time, or construct the element with initial data or children. For example, given the element declaration

```
<!ELEMENT B (#PCDATA | F)*>
```

The following constructors will be provided:

```
B(Document *doc);  
B(Document *doc, String data);  
B(Document *doc, F *theF);
```

The first constructor just makes an empty element with no children. The second initializes it with PCDATA, and the third with a single child node of element F. An element like B which may contain PCDATA is also given a method to add the data post-construction:

```
void addData(Document *doc, String data);
```

The following usages are equivalent:

```
b = new B("data");  
and  
b = new B();  
b->addData("data");
```

Similarly, the following are also equivalent:

```
f = new F(...);  
b = new B(f);  
and  
f = new F(...);  
b = new B();
```

b->addNode(f);

The presence of modifiers '?' (optional), '\*' (zero or more), and '+' (one or more) is ignored when forming the constructors. For example, for the element

```
<!ELEMENT Sample (A* | (B, (C? | (D, E)*)) | F)+>
```

the following constructors are made:

```
Sample(Document *doc);
```

```
Sample(Document *doc, A *theA);
```

```
Sample(Document *doc, B *theB, C *theC);
```

```
Sample(Document *doc, B *theB, D *theD, E *theE);
```

```
Sample(Document *doc, F *theF);
```

as if the modifiers were not present. If you cannot make the desired final element using one of the forms which take initial children, you must start with the empty element and add nodes as needed with addNode as above.

For each attribute for an element, a method is provided to set its value, named setattrname. For example, for the element declaration

```
<!ELEMENT D (#PCDATA)>  
<!ATTLIST D foo CDATA #REQUIRED>
```

the class D will have the method

```
Attr* setfoo(String value);
```

Note: The constructed element is not tested for validity as it is being made. The user to explicitly call the XMLParser's validate method on the final element.

## METHOD INDEX

Constructors	Constructors for the class
addData	Adds PCDATA to the element
addNode	Adds a node to the element
setAttribute	Sets one of the element's attributes



## METHOD

### Constructors

**Function:**

Constructs an element which will belong to the given document. The first form makes the element with no children (use `addData` and `addNode` as appropriate to fill it out). The second variable form is used to provide initial data or children, the exact choices of which depend on the element definition. See the example at the beginning of this document.

**Prototype:**

```
class(Document *doc)  
class(Document *doc, ...)
```

**Arguments:**

`doc` -- document which the element belongs to

`...` -- varying arguments depending on the element definition

**Returns:**

none

### `addData`

**Function:**

Adds data to the element. That is, appends to it a PCDATA subnode with the given value. If multiple `addData` calls are made, the node will have multiple PCDATA subnodes, which should be normalized when construction is finished.

**Prototype:**

```
void addData(Document *doc, String data)
```

**Arguments:**

`doc` -- document which the element belongs to

`data` -- data to be added

**Returns:**

none

**addNode**

**Function:**

Adds (append) a child node to the element. No effort is made to validate the resulting element structure at this time; it is the user's responsibility to form the element properly, which may be verified with `XMLParser::validate`.

**Prototype:**

```
void addNode(node thenode)
```

**Arguments:**

the node -- node to be added

**Returns:**

none

**setAttribute**

**Function:**

Sets the element's attribute with the given value. One method is provided for each attribute, named after the attribute as `setAttribute`.

**Prototype:**

```
Attr* setattribute(String value)
```

**Arguments:**

value -- the attribute's value

**Returns:**

Attr\* -- the created attribute

---

## XML Schema Processor for C++

This chapter describes:

- XML Schema implementation for C++

Note: This beta implementation is incomplete at present; not all schema features are implemented. Also, features can be expected to evolve further before stabilizing.

## Schema APIs

The schema API is very simple: initialize, validate,...validate, terminate.

The validation process is go/no-go. Either the document is valid with respect to the schemas or it is invalid. When it is valid, a zero error code is returned. When it is invalid, a non-zero error code is returned indicating the problem. There is no distinction between warnings and errors; all problems are errors and considered fatal: validation stops immediately.

As schemas are encountered, they are loaded and preserved in the schema context. No schema is loaded more than once during a session. There is no clean up call similar to `xmlclean`. Hence, if you need to release all memory and reset state before validating a new document, you must terminate the context and start over.

## Function/Method Index

**Table 11-1**

Function/Method	Description
<code>XMLSchema::initialize</code>	Initialize the schema processor
<code>XMLSchema::validate</code>	Validate an instance document against a schema
<code>XMLSchema::terminate</code>	Terminates (tears down) the schema processor

## Functions

### XMLSchema::initialize

#### Purpose

Initializes the XML schema processor. Must be called before the processor can be used to validate any documents.

```
uword initialize(xmlctx *ctx)
```

### Parameters

`ctx` (IN) XML parser context

### Comments

The XML parser context is used to allocate memory for the schema context. Note this is not the context of the instance document to be validated.

The context is stored internally and need not be passed around. An error code is returned directly. As always, it is zero on success, non-zero on failure.

## XMLSchema::validate

### Purpose

Validates an instance document against a schema or schemas.

```
uword validate(xmlctx *inst, oratext *schema)
```

### Parameters

`inst` (IN) Instance document context  
`schema` (IN) URL of default schema

### Comments

This is the function which actually performs schema validation. The document to be validated is specified by the XML parser context `inst` used to parse the document. Note the document must already have been parsed.

If no schemas are explicitly referenced in the instance document, the default schema (specified by URL) is assumed. If the document *does* specify all necessary schemas, and a default schema is also supplied, the default will be ignored. If the document does not reference any schemas and no default is supplied, an error will result.

## XMLSchema::terminate

### Purpose

Terminates (shuts down) the schema processor, freeing all memory allocated on the original XML parser context passed to `XMLSchema::Initialize`.

```
void terminate(void)
```

### **Comments**

After termination, the schema context is no longer valid. To continue using the schema processor, a new schema must be created with `XMLSchema::Initialize`.

# Part V

---

## XML-SQL Utility (XSU) Packages

This section contains the following chapters:

- [Chapter 12, "Oracle XML SQL Utility \(XSU\) Java API"](#)
- [Chapter 13, "XML SQL Utility \(XSU\) PL/SQL API"](#)





# 12

---

## Oracle XML SQL Utility (XSU) Java API

## OracleXMLQuery

### Syntax

```
public class OracleXMLQuery extends java.lang.Object
```

```
java.lang.Object
```

```
oracle.xml.sql.query.OracleXMLQuery
```

### Description

The [OracleXMLQuery](#) class does the generation of XML given a SQL query. Following is a very simple example:

```
import java.sql.*;
import oracle.xml.sql.query.*;
import oracle.jdbc.driver.*;

public class sample
{
    public static void main(String args[]) throws Exception
    {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection conn =
            DriverManager.getConnection("jdbc:oracle:oci8:scott/tiger@");
        OracleXMLQuery qry = new OracleXMLQuery(conn, "select * from emp");
        System.out.println(qry.getXMLString());
        conn.close();
    }
}
```

---

### Member Summary

---

#### Fields

##### [DTD](#)

The DTD is used to specified that the DTD is to be generated

##### [ERROR\\_TAG](#)

The ERROR\_TAG specifies the default tag name for the ERROR document

##### [MAXROWS\\_ALL](#)

The MAXROWS\_ALL specifies that all rows be included in the result

---

**Member Summary**


---

<a href="#">NONE</a>	The NONE is used to specified that no DTD is to be generated
<a href="#">ROW_TAG</a>	The ROW_TAG specifies the default tag name for the ROW elements
<a href="#">ROWIDATTR_TAG</a>	The ROWIDATTR_TAG specifies the default tag name for the ROW elements
<a href="#">ROWSET_TAG</a>	The ROWSET_TAG specifies the default tag name for the document
<a href="#">SCHEMA</a>	The SCHEMA is used to specified that no XML schema is to be generated
<a href="#">SKIPROWS_ALL</a>	The SKIPROWS_ALL specifies that all rows be skipped in the result.
Constructors	
<a href="#">OracleXMLQuery(Connection, ResultSet)</a>	Constructor for the OracleXMLQueryObject.
<a href="#">OracleXMLQuery(Connection, String)</a>	Constructor for the OracleXMLQueryObject.
<a href="#">OracleXMLQuery(OracleXMLDataSet)</a>	Constructor for the OracleXMLQueryObject.
Methods	
<a href="#">close()</a>	Close any open resource, created by the OracleXML engine.
<a href="#">getXMLDOM(int)</a>	Transforms the object-relational data, specified in the constructor, into a XML document.
<a href="#">getXMLDOM(Node)</a>	Transforms the object-relational data, specified in the constructor, into XML.
<a href="#">getXMLDOM(Node, int)</a>	Transforms the object-relational data, specified in the constructor, into XML.
<a href="#">getXMLMetaData(int, boolean)</a>	This functions returns the DTD or the XMLSchema for the XML document which would have been generated by a getXML call.
<a href="#">getXMLSAX(ContentHandler)</a>	Transforms the object-relational data, specified in the constructor, into a XML document.
<a href="#">getXMLSchema()</a>	This methods generated the XML Schema(s) corresponding to the specified query.

---

**Member Summary**

---

<a href="#">getXMLString()</a>	Transforms the object-relational data, specified in the constructor, into a XML document.
<a href="#">getXMLString(int)</a>	Transforms the object-relational data, specified in the constructor, into a XML document.
<a href="#">getXMLString(Node)</a>	Transforms the object-relational data, specified in the constructor, into XML.
<a href="#">getXMLString(Node, int)</a>	Transforms the object-relational data, specified in the constructor, into XML.
<a href="#">keepObjectOpen(boolean)</a>	The default behavior for all the <code>getXML</code> functions which DO NOT TAKE in a <code>ResultSet</code> object is to close the <code>ResultSet</code> object and <code>Statement</code> objects at the end of the call.
<a href="#">removeXSLTParam(String)</a>	Removes the value of a top-level stylesheet parameter.
<a href="#">setCollIdAttrName(String)</a>	Sets the name of the id attribute of the collection element's separator tag.
<a href="#">setDataHeader(Reader, String)</a>	Sets the xml data header.
<a href="#">setDateFormat(String)</a>	Sets the format of the generated dates in the XML doc.
<a href="#">setEncoding(String)</a>	Sets the encoding PI (processing instruction) in the XML doc.
<a href="#">setErrorTag(String)</a>	Sets the tag to be used to enclose the xml error docs.
<a href="#">setException(Exception)</a>	Allows the user to pass in an exception, and have the XSU handle it.
<a href="#">setMaxRows(int)</a>	Sets the max number of rows to be converted to XML.
<a href="#">setMetaHeader(Reader)</a>	Sets the XML meta header.
<a href="#">setRaiseException(boolean)</a>	Tells the XSU to throw the raised exceptions.
<a href="#">setRaiseNoRowsException(boolean)</a>	Tells the XSU to throw or not to throw an <code>OracleXMLNoRowsException</code> in the case when for one reason or another, the XML doc generated is empty.
<a href="#">setRowIdAttrName(String)</a>	Sets the name of the id attribute of the row enclosing tag.
<a href="#">setRowIdAttrValue(String)</a>	Specifies the scalar column whose value is to be assigned to the id attribute of the row enclosing tag.
<a href="#">setRowsetTag(String)</a>	Sets the tag to be used to enclose the xml dataset.

---

**Member Summary**


---

<a href="#">setRowTag(String)</a>	Sets the tag to be used to enclose the xml element corresponding to a db.
<a href="#">setSkipRows(int)</a>	Sets the number of rows to skip.
<a href="#">setStylesheetHeader(String)</a>	Sets the stylesheet header (i.e.
<a href="#">setStylesheetHeader(String, String)</a>	Sets the stylesheet header (i.e.
<a href="#">setXSLT(Reader, String)</a>	Registers a XSL transform to be applied to generated XML.
<a href="#">setXSLT(String, String)</a>	Registers a XSL transform to be applied to generated XML.
<a href="#">setXSLTParam(String, String)</a>	Sets the value of a top-level stylesheet parameter.
<a href="#">useLowerCaseTagNames()</a>	This will set the case to be lower for all tag names.
<a href="#">useNullAttributeIndicator(boolean)</a>	Specified weather to use an XML attribute to indicate NULLness, or to do it by omitting the inclusion of the particular entity in the XML document.
<a href="#">useTypeForCollElemTag(boolean)</a>	By default the tag name for elements of a collection is the collection's tag name followed by "_item".
<a href="#">useUpperCaseTagNames()</a>	This will set the case to be upper for all tag names.

---



---

**Inherited Member Summary**


---

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

## Fields

### DTD

```
public static final int DTD
```

The DTD is used to specified that the DTD is to be generated

### ERROR\_TAG

```
public static final java.lang.String ERROR_TAG
```

The ERROR\_TAG specifies the default tag name for the ERROR document

## MAXROWS\_ALL

```
public static final int MAXROWS_ALL
```

The MAXROWS\_ALL specifies that all rows be included in the result

## NONE

```
public static final int NONE
```

The NONE is used to specified that no DTD is to be generated

## ROW\_TAG

```
public static final java.lang.String ROW_TAG
```

The ROW\_TAG specifies the default tag name for the ROW elements

## ROWIDATTR\_TAG

```
public static final java.lang.String ROWIDATTR_TAG
```

The ROWIDATTR\_TAG specifies the default tag name for the ROW elements

## ROWSET\_TAG

```
public static final java.lang.String ROWSET_TAG
```

The ROWSET\_TAG specifies the default tag name for the document

## SCHEMA

```
public static final int SCHEMA
```

The SCHEMA is used to specified that no XML schema is to be generated

## SKIPROWS\_ALL

```
public static final int SKIPROWS_ALL
```

The SKIPROWS\_ALL specifies that all rows be skipped in the result.

## Constructors

### OracleXMLQuery(Connection, ResultSet)

```
public OracleXMLQuery(java.sql.Connection conn, java.sql.ResultSet rset)
```

Constructor for the OracleXMLQueryObject.

#### Parameters:

[conn](#) - database connection

[rset](#) - jdbc result set object

## OracleXMLQuery(Connection, String)

```
public OracleXMLQuery(java.sql.Connection conn, java.lang.String query)
```

Constructor for the OracleXMLQueryObject.

### Parameters:

[conn](#) - database connection

[query](#) - the SQL query string

## OracleXMLQuery(OracleXMLDataSet)

```
public OracleXMLQuery(oracle.xml.sql.dataset.OracleXMLDataSet dset)
```

Constructor for the OracleXMLQueryObject.

### Parameters:

[conn](#) - database connection

[dset](#) - dataset

## Methods

### close()

```
public void close()
```

Close any open resource, created by the OracleXML engine. This will not close for instance resultset supplied by the user

### getNumRowsProcessed()

```
public long getNumRowsProcessed()
```

Returns the number of rows processed.

### Returns:

number of rows processed.

### getXMLDOM()

```
public org.w3c.dom.Document getXMLDOM()
```

Transforms the object-relational data, specified in the constructor, into a XML document.

**Returns:**

the DOM representation of the XML document

**getXMLDOM(int)**

```
public org.w3c.dom.Document getXMLDOM(int metaType)
```

Transforms the object-relational data, specified in the constructor, into a XML document. The [metaType](#) argument is used to specify the type of XML metadata the XSU is to generate along with the XML. Currently this value is ignored, and no XML metadata is generated.

**Parameters:**

[metaType](#) - the type of XML metadata (NONE, SCHEMA)

**Returns:**

the string representation of the XML document

**getXMLDOM(Node)**

```
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root)
```

Transforms the object-relational data, specified in the constructor, into XML. If not NULL, the [root](#) argument, is considered the "root" element of the XML doc.

**Parameters:**

[root](#) - root node to which to append the new XML

**Returns:**

the string representation of the XML document

**getXMLDOM(Node, int)**

```
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root, int metaType)
```

Transforms the object-relational data, specified in the constructor, into XML. If not NULL, the [root](#) argument, is considered the "root" element of the XML doc. The [metaType](#) argument is used to specify the type of XML metadata the XSU is to generate along with the XML. Currently this value is ignored, and no XML metadata is generated.

**Parameters:**

[root](#) - root node to which to append the new XML

[metaType](#) - the type of XML metadata (NONE, SCHEMA)



**Returns:**

the string representation of the XML document

**getXMLMetaData(int, boolean)**

```
public java.lang.String getXMLMetaData(int metaType, boolean withVer)
```

This function returns the DTD or the XMLSchema for the XML document which would have been generated by a getXML call. The "metaType" parameter specifies the type of XML metadata to be generated. The [withVer](#) parameter specifies if version header is to be generated or not.

**Parameters:**

[metaType](#) - XML meta data type to generate (NONE or DTD)

[withVer](#) - generate the version PI ?

**getXMLSAX(ContentHandler)**

```
public void getXMLSAX(org.xml.sax.ContentHandler sax)
```

Transforms the object-relational data, specified in the constructor, into a XML document.

**Parameters:**

[sax](#) - ContentHandler object to be registered

**getXMLSchema()**

```
public org.w3c.dom.Document[] getXMLSchema()
```

This method generates the XML Schema(s) corresponding to the specified query.

**Returns:**

the XML Schema(s)

**getXMLString()**

```
public java.lang.String getXMLString()
```

Transforms the object-relational data, specified in the constructor, into a XML document.

**Returns:**

the string representation of the XML document

## getXMLString(int)

```
public java.lang.String getXMLString(int metaType)
```

Transforms the object-relational data, specified in the constructor, into a XML document.

The [metaType](#) argument is used to specify the type of XML metadata the XSU is to generate along with the XML. Valid values for the [metaType](#) argument are [NONE](#) and [DTD](#) (static fields of this class).

### Parameters:

[metaType](#) - the type of XML metadata (NONE, DTD, or SCHEMA)

### Returns:

the string representation of the XML document

## getXMLString(Node)

```
public java.lang.String getXMLString(org.w3c.dom.Node root)
```

Transforms the object-relational data, specified in the constructor, into XML. If not NULL, the [root](#) argument, is considered the "root" element of the XML doc.

### Parameters:

[root](#) - root node to which to append the new XML

### Returns:

the string representation of the XML document

## getXMLString(Node, int)

```
public java.lang.String getXMLString(org.w3c.dom.Node root, int metaType)
```

Transforms the object-relational data, specified in the constructor, into XML. If not NULL, the [root](#) argument, is considered the "root" element of the XML doc. The [metaType](#) argument is used to specify the type of XML metadata the XSU is to generate along with the XML. Valid values for the [metaType](#) argument are [NONE](#) and [DTD](#) (static fields of this class). Note that if the [root](#) argument is non-null, no DTD is generated even if requested.

### Parameters:

[root](#) - root node to which to append the new XML

[metaType](#) - the type of XML metadata (NONE, DTD, or SCHEMA)

**Returns:**

the string representation of the XML document

**keepObjectOpen(boolean)**

```
public void keepObjectOpen(boolean alive)
```

The default behavior for all the getXML functions which DO NOT TAKE in a ResultSet object is to close the ResultSet object and Statement objects at the end of the call. If you need to use the persistant feature, where by calling getXML repeatedly you get the next set of rows, you need to turn off this behavior by calling this function with value true. i.e. OracleXMLQuery would not close the ResultSet and Statement objects after the getXML calls. You can call the close() function to explicitly close the cursor state.

**Parameters:**

[alive](#) - keep object open ?

**removeXSLTParam(String)**

```
public void removeXSLTParam(java.lang.String name)
```

Removes the value of a top-level stylesheet parameter. NOTE: if no stylesheet is registered, this method is a no op.

**Parameters:**

[name](#) - parameter name

**setCollIdAttrName(String)**

```
public void setCollIdAttrName(java.lang.String attrName)
```

Sets the name of the id attribute of the collection element's separator tag. Passing [null](#) or an empty string for the [tag](#) results the row id attribute to be omitted.

**Parameters:**

[attrName](#) - attribute name

**setDataHeader(Reader, String)**

```
public void setDataHeader(java.io.Reader header, java.lang.String docTag)
```

Sets the xml data header. The data header is an XML entity which is appended at the beginning of the query-generated xml entity (ie. rowset). The two entities are enclosed by the tag specified via the [docTag](#) argument. Note that the last data header specified is the one that is used; furthermore, passing in [null](#) for the [header](#), parameter unsets the data header.

**Parameters:**

[header](#) - header

[tag](#) - tag used to enclose the data header and the rowset

**setDateFormat(String)**

```
public void setDateFormat(java.lang.String mask)
```

Sets the format of the generated dates in the XML doc. The syntax of the date format pattern (i.e. the date mask), should conform to the requirements of the `java.text.SimpleDateFormat` class. Setting the mask to [null](#) or an empty string, unsets the date mask.

**Parameters:**

[mask](#) - the date mask

**setEncoding(String)**

```
public void setEncoding(java.lang.String enc)
```

Sets the encoding PI (processing instruction) in the XML doc. If [null](#) or an empty string are specified as the encoding, then the default charsetset is specified in the encoding PI.

**Parameters:**

[enc](#) - charsetset encoding of the XML doc

**setErrorTag(String)**

```
public void setErrorTag(java.lang.String tag)
```

Sets the tag to be used to enclose the xml error docs.

**Parameters:**

[tag](#) - tag name

**setException(Exception)**

```
public void setException(java.lang.Exception e)
```

Allows the user to pass in an exception, and have the XSU handle it.

**Parameters:**

[e](#) - the exception to be processed by the XSU.

## setMaxRows(int)

```
public void setMaxRows(int rows)
```

Sets the max number of rows to be converted to XML. By default there is no max set. To explicitly specify no max see MAXROWS\_ALL.

### Parameters:

[rows](#) - max number of rows to generate

## setMetaHeader(Reader)

```
public void setMetaHeader(java.io.Reader header)
```

Sets the XML meta header. When set, the header is inserted at the beginning of the metadata part (DTD or XMLSchema) of each XML document generated by this object. Note that the last meta header specified is the one that is used; furthermore, passing in [null](#) for the [header](#), parameter unsets the meta header.

### Parameters:

[header](#) - header

## setRaiseException(boolean)

```
public void setRaiseException(boolean flag)
```

Tells the XSU to throw the raised exceptions. If this call isn't made or if [false](#) is passed to the [flag](#) argument, the XSU catches the SQL exceptions and generates an XML doc out of the exception's message.

### Parameters:

[flag](#) - throw raised exceptions?

## setRaiseNoRowsException(boolean)

```
public void setRaiseNoRowsException(boolean flag)
```

Tells the XSU to throw or not to throw an OracleXMLNoRowsException in the case when for one reason or another, the XML doc generated is empty. By default, the exception is not thrown.

### Parameters:

[flag](#) - throw OracleXMLNoRowsException if no data found?

## setRowIdAttrName(String)

```
public void setRowIdAttrName(java.lang.String attrName)
```

Sets the name of the id attribute of the row enclosing tag. Passing [null](#) or an empty string for the [tag](#) results the row id attribute to be omitted.

### Parameters:

[attrName](#) - attribute name

## setRowIdAttrValue(String)

```
public void setRowIdAttrValue(java.lang.String colName)
```

Specifies the scalar column whose value is to be assigned to the id attribute of the row enclosing tag. Passing [null](#) or an empty string for the [colName](#) results the row id attribute being assigned the row count value (i.e. 0, 1, 2 and so on).

### Parameters:

[colName](#) - column whose value is to be assigned to the row id attr

## setRowsetTag(String)

```
public void setRowsetTag(java.lang.String tag)
```

Sets the tag to be used to enclose the xml dataset.

### Parameters:

[tag](#) - tag name

## setRowTag(String)

```
public void setRowTag(java.lang.String tag)
```

Sets the tag to be used to enclose the xml element corresponding to a db. record.

### Parameters:

[tag](#) - tag name

## setSkipRows(int)

```
public void setSkipRows(int rows)
```

Sets the number of rows to skip. By default 0 rows are skipped. To skip all the rows use SKIPROWS\_ALL.

**Parameters:**

[rows](#) - number of rows to skip

**setStylesheetHeader(String)**

```
public void setStylesheetHeader(java.lang.String uri)
```

Sets the stylesheet header (i.e. stylesheet processing instructions) in the generated XML doc.

Note: Passing [null](#) for the [uri](#) argument will unset the stylesheet header and the stylesheet type.

**Parameters:**

[uri](#) - stylesheet URI

**setStylesheetHeader(String, String)**

```
public void setStylesheetHeader(java.lang.String uri, java.lang.String type)
```

Sets the stylesheet header (i.e. stylesheet processing instructions) in the generated XML doc.

Note: Passing [null](#) for the [uri](#) argument will unset the stylesheet header and the stylesheet type.

**Parameters:**

[uri](#) - stylesheet URI

[type](#) - stylesheet type; defaults to 'text/xsl'

**setXSLT(Reader, String)**

```
public void setXSLT(java.io.Reader stylesheet, java.lang.String ref)
```

Registers a XSL transform to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a [null](#) for the [stylesheet](#) argument.

**Parameters:**

[stylesheet](#) - the stylesheet

[ref](#) - URL for include, import and external entities

**setXSLT(String, String)**

```
public void setXSLT(java.lang.String stylesheet, java.lang.String ref)
```

Registers a XSL transform to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a [null](#) for the [stylesheet](#) argument.

**Parameters:**

[stylesheet](#) - the stylesheet URI

[ref](#) - URL for include, import and external entities

**setXSLTParam(String, String)**

```
public void setXSLTParam(java.lang.String name, java.lang.String value)
```

Sets the value of a top-level stylesheet parameter. The parameter value is expected to be a valid XPath expression (note that string literal values would therefore have to be explicitly quoted). NOTE: if no stylesheet is registered, this method is a no op.

**Parameters:**

[name](#) - parameter name

[value](#) - parameter value as an XPATH expression

**useLowerCaseTagNames()**

```
public void useLowerCaseTagNames()
```

This will set the case to be lower for all tag names. Note, make this call after all the desired tags have been set.

**useNullAttributeIndicator(boolean)**

```
public void useNullAttributeIndicator(boolean flag)
```

Specified whether to use an XML attribute to indicate NULLness, or to do it by omitting the inclusion of the particular entity in the XML document.

**Parameters:**

[flag](#) - use attribute to indicate null?

**useTypeForCollElemTag(boolean)**

```
public void useTypeForCollElemTag(boolean flag)
```

By default the tag name for elements of a collection is the collection's tag name followed by "\_item". This method, when called with argument of [true](#), tells the XSU to use the collection element's type name as the collection element tag name.

**Parameters:**

[flag](#) - use the coll. elem. type as its tag name?



## **useUpperCaseTagNames()**

```
public void useUpperCaseTagNames()
```

This will set the case to be upper for all tag names. Note, make this call after all the desired tags have been set.

## OracleXMLSave

### Syntax

```
public class OracleXMLSave extends java.lang.Object

java.lang.Object
|
+--oracle.xml.sql.dml.OracleXMLSave
```

### Description

OracleXMLSave - this class supports canonical mapping from XML to object-relational tables or views. It supports inserts, updates and deletes. The user first creates the class by passing in the table name on which these DML operations need to be done. After that, the user is free to use the insert/update/delete on this table. A typical sequence might look like

```
OracleXMLSave sav = new OracleXMLSave(conn, "emp");
// insert processing
sav.insertXML(xmlDoc);
-or-
// Update processing
String[]tempArr = new String[2];
tempArr[0] = "EMPNO"; // set the empno as the key for updates..
sav.setKeyColumnList(tempArray);
sav.updateXML(xmlDoc);
-or-
sav.deleteXML(xmlDoc);
sav.close();
```

A lot of useful functions are provided in this class to help in identifying the key columns for update or delete and to restrict the columns being updated.

---

## Member Summary

---

### Fields

<a href="#">DATE_FORMAT</a>	The date format for use in setDateFormat
<a href="#">DEFAULT_BATCH_SIZE</a>	default insert batch size is 17

### Constructors

<a href="#">OracleXMLSave(Connection, String)</a>	The public constructor for the Save class.
---	--

### Methods

<a href="#">cleanLobList()</a>	
<a href="#">close()</a>	It closes/deallocates all the context associated with this object.
<a href="#">deleteXML(Document)</a>	Deletes the rows in the table based on the XML document
<a href="#">deleteXML(InputStream)</a>	Deletes the rows in the table based on the XML document
<a href="#">deleteXML(Reader)</a>	Deletes the rows in the table based on the XML document
<a href="#">deleteXML(String)</a>	Deletes the rows in the table based on the XML document
<a href="#">deleteXML(URL)</a>	Deletes rows from a specified table based on the element values in the supplied XML document.
<a href="#">finalize()</a>	
<a href="#">getURL(String)</a>	Given a file name or a URL it return a URL object.
<a href="#">insertXML(Document)</a>	
<a href="#">insertXML(InputStream)</a>	
<a href="#">insertXML(Reader)</a>	
<a href="#">insertXML(String)</a>	
<a href="#">insertXML(URL)</a>	Inserts an XML document from a specified URL into the specified table, By default, the insert routine inserts the values into the table by matching the element name with the column name and inserts a null value for all elements that are missing in the input document.

---

**Member Summary**

---

<a href="#">removeXSLTParam(String)</a>	Removes the value of a top-level stylesheet parameter.
<a href="#">setBatchSize(int)</a>	This call changes the batch size used during DML operations.
<a href="#">setCommitBatch(int)</a>	Sets the commit batch size.
<a href="#">setDateFormat(String)</a>	Describes to the XSU the format of the dates in the XML document.
<a href="#">setIgnoreCase(boolean)</a>	The XSU does mapping of XML elements to database columns/attrs.
<a href="#">setKeyColumnList(String[])</a>	Sets the list of columns to be used for identifying a particular row in the database table during update or delete.
<a href="#">setRowTag(String)</a>	Names the tag used in the XML doc., to enclose the XML elements corresponding to each row value.
<a href="#">setUpdateColumnList(String[])</a>	Set the column values to be updated.
<a href="#">setXSLT(Reader, String)</a>	Registers a XSL transform to be applied to generated XML.
<a href="#">setXSLT(String, String)</a>	Registers a XSL transform to be applied to generated XML.
<a href="#">setXSLTParam(String, String)</a>	Sets the value of a top-level stylesheet parameter.
<a href="#">updateXML(Document)</a>	Updates the table given the XML document in a DOM tree form
<a href="#">updateXML(InputStream)</a>	Updates the table given the XML document in a stream form
<a href="#">updateXML(Reader)</a>	Updates the table given the XML document in a stream form
<a href="#">updateXML(String)</a>	Updates the table given the XML document in a string form
<a href="#">updateXML(URL)</a>	Updates the columns in a database table, based on the element values in the supplied XML document.

---

---

**Inherited Member Summary**

---

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

## Fields

### DATE\_FORMAT

```
public static final java.lang.String DATE_FORMAT
```

The date format for use in setDateFormat

### DEFAULT\_BATCH\_SIZE

```
public static int DEFAULT_BATCH_SIZE
```

default insert batch size is 17

## Constructors

### OracleXMLSave(Connection, String)

```
public OracleXMLSave(java.sql.Connection oconn, java.lang.String tableName)
```

The public constructor for the Save class.

#### Parameters:

[oconn](#) - Connection object (connection to the database)

[tableName](#) - The name of the table that should be updated

## Methods

### cleanLobList()

```
public void cleanLobList()
```

### close()

```
public void close()
```

It closes/deallocates all the context associated with this object.

### deleteXML(Document)

```
public int deleteXML(org.w3c.dom.Document doc)
```

Deletes the rows in the table based on the XML document

#### Parameters:

[xmlDoc](#) - The XML document in DOM form

**Returns:**

The number of XML ROW elements processed.

**See Also:**

[deleteXML\(URL\)](#)

**deleteXML(InputStream)**

```
public int deleteXML(java.io.InputStream xmlStream)
```

Deletes the rows in the table based on the XML document

**Parameters:**

[xmlDoc](#) - The XML document in Stream form

**Returns:**

The number of XML ROW elements processed.

**See Also:**

[deleteXML\(URL\)](#)

**deleteXML(Reader)**

```
public int deleteXML(java.io.Reader xmlStream)
```

Deletes the rows in the table based on the XML document

**Parameters:**

[xmlDoc](#) - The XML document in Stream form

**Returns:**

The number of XML ROW elements processed.

**See Also:**

[deleteXML\(URL\)](#)

**deleteXML(String)**

```
public int deleteXML(java.lang.String xmlDoc)
```

Deletes the rows in the table based on the XML document

**Parameters:**

[xmlDoc](#) - The XML document in String form

**Returns:**

The number of XML ROW elements processed.

**See Also:**

[deleteXML\(URL\)](#)

**deleteXML(URL)**

```
public int deleteXML(java.net.URL url)
```

Deletes rows from a specified table based on the element values in the supplied XML document. By default, the delete processing matches all the element values with the corresponding column names. Each ROW element in the input document is taken as a separate delete statement on the table. By using the `setKeyColumnList()` you can set the list of columns that must be matched to identify the row to be deleted and ignore the other elements. This is an efficient method for deleting more than one row in the table, since the delete statement is cached and batching can be employed. If not, a new delete statement has to be created for each ROW element in the input document.

For example, consider the employee table emp,

```
TABLE emp( empno NUMBER, ename VARCHAR2(20), hiredate DATE);
```

Now, you want to delete the rows in the table using an XML document and you want to identify the row based on the EMPNO value. You can send in an XML document containing the EMPNO value and call the `deleteXML` routine with the `setKeyColumnList()` containing the EMPNO string.

```
OracleXMLSave save = new OracleXMLSave(conn,"emp");
```

```
save.deleteXML(xmlDoc); // Deletes rows by matching all columns with the // element
values in the input document.
```

```
String[] deleteArray = new String[1];
```

```
deleteArray[0] = "EMPNO";
```

```
save.setKeyColumnList(deleteArray); // set the key columns
```

```
save.deleteXML(xmlDoc); // only deletes rows by matching the EMPNO values
```

**Parameters:**

[url](#) - The URL to the document to use to delete the rows in the table

**Returns:**

The number of XML row elements processed. Note that this may or may not be equal to the number of database rows deleted based on whether the rows selected through the XML document uniquely identified the rows in the table.

**finalize()**

```
protected void finalize()
```

**Overrides:**

java.lang.Object.finalize() in class java.lang.Object

**getURL(String)**

```
public static java.net.URL getURL(java.lang.String target)
```

Given a file name or a URL it return a URL object. If the argument passed is not in the valid URL format (e.g. http://.. or file://) then this method tried to fix the argument by pre-pending "file://" to the argument. If a [null](#) or an empty string are passed to it, null is returned.

**Parameters:**

[target](#) - file name or URL string

**Returns:**

the URL object identifying the [target entity](#)

**insertXML(Document)**

```
public int insertXML(org.w3c.dom.Document doc)
```

**insertXML(InputStream)**

```
public int insertXML(java.io.InputStream xmlStream)
```

**insertXML(Reader)**

```
public int insertXML(java.io.Reader xmlStream)
```

**insertXML(String)**

```
public int insertXML(java.lang.String xmlDoc)
```



## insertXML(URL)

```
public int insertXML(java.net.URL url)
```

Inserts an XML document from a specified URL into the specified table. By default, the insert routine inserts the values into the table by matching the element name with the column name and inserts a null value for all elements that are missing in the input document. By setting the list of columns to insert using the `setUpdateColumnList()` you can restrict the insert to only insert values into those columns and let the default values for other columns to be inserted. That is no null value would be inserted for the rest of the columns. Use `setKeyColumnList()` to set the list of all key column. Use `setUpdateColumnList()` to set the list of columns to update.

For example, consider the employee table emp,

```
TABLE emp( empno NUMBER, ename VARCHAR2(20), hiredate DATE);
```

Now, assume that you want to insert the table using an XML document OracleXMLSave

```
save = new OracleXMLSave(conn,"emp");
save.insertXML(xmlDoc); // xmlDoc supplied..
save.close();
```

If you want to insert values only in to EMPNO, HIREDATE and SALARY and let the default values handle the rest of the columns,

```
String insArray = new String[3];
insArray[0] = "EMPNO";
insArray[1] = "HIREDATE";
insArray[2] = "SALARY";
save.setUpdateColumnList(insArray);
save.insertXML(xmlDoc); // will only insert values into EMPNO, HIREDATE // and
SALARY columns
```

### Parameters:

[url](#) - The URL to the document to use to insert rows into the table

### Returns:

The number of rows inserted.

## removeXSLTParam(String)

```
public void removeXSLTParam(java.lang.String name)
```

Removes the value of a top-level stylesheet parameter. NOTE: if no stylesheet is registered, this method is a no op.

### Parameters:

[name](#) - parameter name

## setBatchSize(int)

```
public void setBatchSize(int size)
```

This call changes the batch size used during DML operations. When performing inserts, updates or deletes, it is better to batch the operations so that the database can execute it in one shot rather than as separate statements. The flip side is that more memory is needed to hold all the bind values before the operation is done. Note that when batching is used, the commits occur only in terms of batches. So if one of the statement inside a batch fails, the whole batch is rolled back. If this behaviour is unacceptable, then set the batch size to 1. The default batch size is `DEFAULT_BATCH_SIZE`;

### Parameters:

[size](#) - The batch size to use for all DML

## setCommitBatch(int)

```
public void setCommitBatch(int size)
```

Sets the commit batch size. The commit batch size refers to the number of records inserted after which a commit should follow. Note that if `commitBatch` is `< 1` or the session is in "auto-commit" mode then the XSU does not make any explicit commit's. By default the commit-batch size is 0.

### Parameters:

[size](#) - commit batch size

## setDateFormat(String)

```
public void setDateFormat(java.lang.String mask)
```

Describes to the XSU the format of the dates in the XML document. By default, OracleXMLSave assumes that the date is in format 'MM/dd/yyyy HH:mm:ss'. You can override this default format by calling this function. The syntax of the date format pattern (i.e. the date mask), should conform to the requirements of the `java.text.SimpleDateFormat`

class. Setting the mask to [null](#) or an empty string, results the use of the default mask -- OracleXMLSave.DATE\_FORMAT.

**Parameters:**

[mask](#) - the date mask

**setIgnoreCase(boolean)**

```
public void setIgnoreCase(boolean ignore)
```

The XSU does mapping of XML elements to database columns/attrs. based on the element names (xml tags). This function tells the XSU to do this match case insensitive. This resetting of case may affect the metadata caching that is done when creating the Save object.

**Parameters:**

[flag](#) - ignore tag case in the XML doc? 0-false 1-true

**setKeyColumnList(String[])**

```
public void setKeyColumnList(java.lang.String[] keyColNames)
```

Sets the list of columns to be used for identifying a particular row in the database table during update or delete. This call is ignored for the insert case. The key columns must be set before updates can be done. It is optional for deletes. When this key columns is set, then the values from these tags in the XML document is used to identify the database row for update or delete. Currently, there is no way to update the values of the key columns themselves, since there is no way in the XML document to specify that case

**Parameters:**

[keyColNames](#) - The names of the list of columns that are used as keys

**setRowTag(String)**

```
public void setRowTag(java.lang.String rowTag)
```

Names the tag used in the XML doc., to enclose the XML elements corresponding to each row value. Setting the value of this to null implies that there is no row tag present and the top level elements of the document correspond to the rows themselves.

**Parameters:**

[tag](#) - tag name

## setUpdateColumnList(String[])

```
public void setUpdateColumnList(java.lang.String[] updColNames)
```

Set the column values to be updated. This is only valid for inserts and updates, and is ignored for deletes. In case of insert, the default is to insert values to all the columns in the table. In case of updates, the default is to only update the columns corresponding to the tags present in the ROW element of the XML document. When specified, these columns alone will get updated in the update or insert statement. All other elements in the document will be ignored.

### Parameters:

[updColNames](#) - The string list of columns to be updated

## setXSLT(Reader, String)

```
public void setXSLT(java.io.Reader stylesheet, java.lang.String ref)
```

Registers a XSL transform to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a [null](#) for the [stylesheet](#) argument.

### Parameters:

[stylesheet](#) - the stylesheet

[ref](#) - URL for include, import and external entities

## setXSLT(String, String)

```
public void setXSLT(java.lang.String stylesheet, java.lang.String ref)
```

Registers a XSL transform to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a [null](#) for the [stylesheet](#) argument.

### Parameters:

[stylesheet](#) - the stylesheet URI

[ref](#) - URL for include, import and external entities

## setXSLTParam(String, String)

```
public void setXSLTParam(java.lang.String name, java.lang.String value)
```

Sets the value of a top-level stylesheet parameter. The parameter value is expected to be a valid XPath expression (note that string literal values would therefore have to be explicitly quoted). NOTE: if no stylesheet is registered, this method is a no op.

**Parameters:**

[name](#) - parameter name

[value](#) - parameter value as an XPATH expression

**updateXML(Document)**

```
public int updateXML(org.w3c.dom.Document doc)
```

Updates the table given the XML document in a DOM tree form

**Parameters:**

[xmlDoc](#) - The DOM tree form of the XML document

**Returns:**

The number of XML elements processed

**See Also:**

[updateXML\(URL\)](#)

**updateXML(InputStream)**

```
public int updateXML(java.io.InputStream xmlStream)
```

Updates the table given the XML document in a stream form

**Parameters:**

[xmlDoc](#) - The stream form of the XML document

**Returns:**

The number of XML elements processed

**See Also:**

[updateXML\(URL\)](#)

**updateXML(Reader)**

```
public int updateXML(java.io.Reader xmlStream)
```

Updates the table given the XML document in a stream form

**Parameters:**

[xmlDoc](#) - The stream form of the XML document

**Returns:**

The number of XML elements processed

**See Also:**

[updateXML\(URL\)](#)

**updateXML(String)**

```
public int updateXML(java.lang.String xmlDoc)
```

Updates the table given the XML document in a string form

**Parameters:**

[xmlDoc](#) - The string form of the XML document

**Returns:**

The number of XML elements processed

**See Also:**

[updateXML\(URL\)](#)

**updateXML(URL)**

```
public int updateXML(java.net.URL url)
```

Updates the columns in a database table, based on the element values in the supplied XML document. The update requires a list of key columns which are used to uniquely identify a row to update in the given table. By default, the update uses the list of key columns and matches the values of the corresponding elements in the XML document to identify a particular row and then updates all the columns in the table for which there is an equivalent element present in the XML document.

Each ROW element present in the input document is treated as a separate update to the table.

You can also supply a list of columns to update - this will make the utility update only those columns and ignore any other element present in the XML document. This is a very efficient method, since if there are more than one row present in the input XML document, the update statement itself is cached and batching is done. If not, then we need to create a new update statement for each row of the input document.

Use `setKeyColumnList()` to set the list of all key column.

Use `setUpdateColumnList()` to set the list of columns to update.

For example, consider the employee table emp,

```
TABLE emp( empno NUMBER, ename VARCHAR2(20), hiredate DATE);
```

Now, assume that you want to update the table using an XML document and you want to use the values of the element EMPNO to match the right row and then only update the HIREDATE column.

You can send in an XML document containing the HIREDATE value and the EMPNO value and call the updateXML routine with the setKeyColumnList() containing the EMPNO string.

```
OracleXMLSave save = new OracleXMLSave(conn,"emp");
String[] keyArray = new String[1];
keyArray[0] = "EMPNO"; // Set EMPNO as key column
save.setKeyColumnList(keyArray); // Set the key column names
String[] updArray = new String[1];
updArray[0] = "HIREDATE"; // SEt hiredate as column to update
save.setUpdateColumnList(updArray);
save.updateXML(xmlDoc); // xmlDoc supplied..
save.close();
```

**Parameters:**

[url](#) - The URL to the document to use to update the table

**Returns:**

The number of XML row elements processed. Note that this may or may not be equal to the number of database rows modified based on whether the rows selected through the XML document uniquely identified the rows in the table.

# OracleXMLSQLException

## Syntax

```
public class OracleXMLSQLException extends java.lang.RuntimeException
```

```
java.lang.Object
|
+--java.lang.Throwable
   |
   +--java.lang.Exception
      |
      +--java.lang.RuntimeException
         |
         +--oracle.xml.sql.OracleXMLSQLException
```

## Direct Known Subclasses:

[OracleXMLSQLNoRowsException](#)

## All Implemented Interfaces:

[java.io.Serializable](#)

## Description

---

### Member Summary

---

#### Constructors

[OracleXMLSQLException\(Exception\)](#)

[OracleXMLSQLException\(Exception, String\)](#)

[OracleXMLSQLException\(String\)](#)

[OracleXMLSQLException\(String, Exception\)](#)

[OracleXMLSQLException\(String, Exception, String\)](#)

[OracleXMLSQLException\(String, int\)](#)

[OracleXMLSQLException\(String, int, String\)](#)

[OracleXMLSQLException\(String, String\)](#)

#### Methods



---

**Member Summary**

---

<a href="#">getErrorCode()</a>	
<a href="#">getParentException()</a>	returns the original exception, if there was one; otherwise, it returns null
<a href="#">getXMLErrorString()</a>	prints the XML error string with the given error tag name
<a href="#">getXMLSQLExceptionString()</a>	prints the SQL parameters as well in the error message
<a href="#">setErrorTag(String)</a>	Sets the error tag name which is then used by <a href="#">getXMLErrorString</a> and <a href="#">getXMLSQLExceptionString</a> , to generate xml error reports

---



---

**Inherited Member Summary**

---

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace`, `getLocalizedMessage`, `getMessage`, `printStackTrace`, `printStackTrace`, `printStackTrace`, `toString`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

---

## Constructors

### OracleXMLSQLException(Exception)

```
public OracleXMLSQLException(java.lang.Exception e)
```

### OracleXMLSQLException(Exception, String)

```
public OracleXMLSQLException(java.lang.Exception e, java.lang.String errorTagName)
```

### OracleXMLSQLException(String)

```
public OracleXMLSQLException(java.lang.String message)
```

### OracleXMLSQLException(String, Exception)

```
public OracleXMLSQLException(java.lang.String message, java.lang.Exception e)
```

## OracleXMLSQLException(String, Exception, String)

```
public OracleXMLSQLException(java.lang.String message, java.lang.Exception e,  
java.lang.String errorTagName)
```

## OracleXMLSQLException(String, int)

```
public OracleXMLSQLException(java.lang.String message, int errorCode)
```

## OracleXMLSQLException(String, int, String)

```
public OracleXMLSQLException(java.lang.String message, int errorCode,  
java.lang.String errorTagName)
```

## OracleXMLSQLException(String, String)

```
public OracleXMLSQLException(java.lang.String message, java.lang.String  
errorTagName)
```

## Methods

### getErrorCode()

```
public int getErrorCode()
```

### getParentException()

```
public java.lang.Exception getParentException()  
returns the original exception, if there was one; otherwise, it returns null
```

### getXMLErrorString()

```
public java.lang.String getXMLErrorString()  
prints the XML error string with the given error tag name
```

### getXMLSQLExceptionString()

```
public java.lang.String getXMLSQLExceptionString()  
prints the SQL parameters as well in the error message
```

### setErrorTag(String)

```
public void setErrorTag(java.lang.String tagName)  
Sets the error tag name which is then used by getXMLErrorString and  
getXMLSQLExceptionString, to generate xml error reports
```

# OracleXMLSQLNoRowsException

## Syntax

public class OracleXMLSQLNoRowsException extends [OracleXMLSQLException](#)

```

java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.lang.RuntimeException
            |
            +--OracleXMLSQLException
                |
                +--oracle.xml.sql.OracleXMLSQLNoRowsException
  
```

## All Implemented Interfaces:

java.io.Serializable

---

## Member Summary

Constructors

[OracleXMLSQLNoRowsException\(\)](#)

[OracleXMLSQLNoRowsException\(String\)](#)

---

## Inherited Member Summary

Methods inherited from interface [OracleXMLSQLException](#)

[getErrorCode\(\)](#), [getParentException\(\)](#), [getXMLErrorMessage\(\)](#), [getXMLSQLExceptionString\(\)](#), [setErrorTag\(String\)](#)

Methods inherited from class java.lang.Throwable

[fillInStackTrace\(\)](#), [getLocalizedMessage\(\)](#), [getMessage\(\)](#), [printStackTrace\(\)](#), [printStackTrace\(PrintWriter\)](#), [toString\(\)](#)

Methods inherited from class java.lang.Object

[clone\(\)](#), [equals\(Object\)](#), [finalize\(\)](#), [getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(long\)](#), [wait\(long, int\)](#)

## Constructors

### OracleXMLSQLNoRowsException()

```
public OracleXMLSQLNoRowsException()
```

### OracleXMLSQLNoRowsException(String)

```
public OracleXMLSQLNoRowsException(java.lang.String errorTag)
```

---

## XML SQL Utility (XSU) PL/SQL API

XSU's PL/SQL API consists of two packages:

- `DBMS_XMLQuery` -- provides `DB_to_XML` type functionality.
- `DBMS_XMLSave` -- provides `XML_to_DB` type functionality.

## DBMS\_XMLQuery

### Types:

#### ctxType

The type of the query context handle. This is the return type of "DBMS\_XMLQuery.newContext()".

### Constants:

#### DEFAULT\_ROWSETTAG

The tag name for the element enclosing the XML generated from the result set (i.e. for most cases the root node tag name) -- ROWSET

#### DEFAULT\_ERRORTAG

The default tag to enclose raised errors -- ERROR.

#### DEFAULT\_ROWIDATTR

The default name for the cardinality attribute of XML elements corresponding to db. records. -- NUM

#### DEFAULT\_ROWTAG

The default tag name for the element corresponding to db. records. -- ROW

#### DEFAULT\_DATE\_FORMAT

Default date mask. -- 'MM/dd/yyyy HH:mm:ss'

#### ALL\_ROWS

The ALL\_ROWS parameter is to indicate that all rows are needed in the output.

#### NONE

Used to specify that the output should not contain any XML metadata (e.g. no DTD).

#### DTD

Used to specify that the generation of the DTD is desired.

#### SCHEMA

Used to specify that the generation of the XML SCHEMA is desired.

**LOWER\_CASE**

Use lower cased tag names.

**UPPER\_CASE**

Use upper case tag names.

**Function and Procedure Index:****PROCEDURE closeContext(ctxType)**

It closes/deallocates a particular query context

**FUNCTION getDTD(ctxType, BOOLEAN := false) RETURN CLOB**

Generates the DTD based on the SQL query used to init.

**PROCEDURE getDTD(ctxType, CLOB, BOOLEAN := false)**

Generates the DTD based on the SQL query used to init.

**PROCEDURE getExceptionContent(ctxType, NUMBER, VARCHAR2)**

Via its arguments, this method returns the thrown exception's error code and error message (i.e.

**FUNCTION getXML(VARCHAR2, NUMBER := NONE) RETURN CLOB**

Generates the XML doc.

**FUNCTION getXML(CLOB, NUMBER := NONE) RETURN CLOB**

Generates the XML doc.

**FUNCTION getXML(ctxType, NUMBER := NONE) RETURN CLOB**

Generates the XML doc.

**PROCEDURE getXML(ctxType, CLOB, NUMBER := NONE)**

Generates the XML doc.

**FUNCTION newContext(VARCHAR2) RETURN ctxType**

It creates a query context, and it returns the context handle.

**FUNCTION newContext(CLOB) RETURN ctxType**

It creates a query context, and it returns the context handle.

**PROCEDURE propagateOriginalException(ctxType, BOOLEAN)**

Tells the XSU that if an exception is raised, and is being thrown, the XSU should throw the very exception raised; rather than, wrapping it with an OracleXMLSQLException.

**PROCEDURE setBindValue(ctxType, VARCHAR2, VARCHAR2)**

Sets a value for a particular bind name.

**PROCEDURE setCollIdAttrName(ctxType, VARCHAR2)**

Sets the name of the id attribute of the collection element's separator tag.

**PROCEDURE setDataHeader(ctxType, CLOB := null, VARCHAR2 := null)**

Sets the xml data header.

**PROCEDURE setDateFormat(ctxType, VARCHAR2)**

Sets the format of the generated dates in the XML doc.

**PROCEDURE setErrorTag(ctxType, VARCHAR2)**

Sets the tag to be used to enclose the xml error docs.

**PROCEDURE setMaxRows (ctxType, NUMBER)**

Sets the max number of rows to be converted to XML.

**PROCEDURE setMetaHeader(ctxType, CLOB := null)**

Sets the XML meta header.

**PROCEDURE setRaiseException(ctxType, BOOLEAN)**

Tells the XSU to throw the raised exceptions.

**PROCEDURE setRaiseNoRowsException(ctxType, BOOLEAN)**

Tells the XSU to throw or not to throw an OracleXMLNoRowsException in the case when for one reason or another, the XML doc generated is empty.

**PROCEDURE setRowIdAttrName(ctxType, VARCHAR2)**

Sets the name of the id attribute of the row enclosing tag.

**PROCEDURE setRowIdAttrValue(ctxType, VARCHAR2)**

Specifies the scalar column whose value is to be assigned to the id attribute of the row enclosing tag.

**PROCEDURE setRowsetTag(ctxType, VARCHAR2)**

Sets the tag to be used to enclose the xml dataset.

**PROCEDURE setRowTag(ctxType, VARCHAR2)**

Sets the tag to be used to enclose the xml element corresponding to a db.

**PROCEDURE setSkipRows(ctxType, NUMBER)**

Sets the number of rows to skip.

**PROCEDURE setStylesheetHeader(ctxType, VARCHAR2, VARCHAR2 := 'text/xsl')**

Sets the stylesheet header (i.e.

**PROCEDURE setTagCase(ctxType, NUMBER)**

Specified the case of the generated XML tags.

**PROCEDURE setXSLT(ctxType, VARCHAR2, VARCHAR2 := null)**

Registers a stylesheet to be applied to generated XML.



**PROCEDURE setXSLT(ctxType, CLOB, VARCHAR2 := null)**

Registers a stylesheet to be applied to generated XML.

**PROCEDURE setXSLTParam(ctxType, VARCHAR2, VARCHAR2)**

Sets the value of a top-level stylesheet parameter.

**PROCEDURE removeXSLTParam(ctxType, VARCHAR2)**

Removes a particular top-level stylesheet parameter.

**PROCEDURE useNullAttributeIndicator(ctxType, BOOLEAN)**

Specifies whether to use an XML attribute to indicate NULLness, or to do it by omitting the inclusion of the particular entity in the XML document.

**Functions and Procedures:****newContext**

FUNCTION newContext(sqlQuery IN VARCHAR2) RETURN ctxType

It creates a query context, and it returns the context handle.

**Parameters:**

sqlQuery - SQL query, the results of which to convert to XML

**Returns:**

The context handle.

**newContext**

FUNCTION newContext(sqlQuery IN CLOB) RETURN ctxType

It creates a query context, and it returns the context handle.

**Parameters:**

sqlQuery - SQL query, the results of which to convert to XML

**Returns:**

The context handle.

**closeContext**

PROCEDURE closeContext(ctxHdl IN ctxType)

It closes/deallocates a particular query context

**Parameters:**

ctxHdl - context handle

**setRowsetTag**

PROCEDURE setRowsetTag(ctxHdl IN ctxType, tag IN VARCHAR2)

Sets the tag to be used to enclose the xml dataset.

**Parameters:**

ctxHdl - context handle

tag - tag name

### **setRowTag**

PROCEDURE setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2)

Sets the tag to be used to enclose the xml element corresponding to a db. record.

#### **Parameters:**

ctxHdl - context handle

tag - tag name

### **setErrorTag**

PROCEDURE setErrorTag(ctxHdl IN ctxType, tag IN VARCHAR2)

Sets the tag to be used to enclose the xml error docs.

#### **Parameters:**

ctxHdl - context handle

tag - tag name

### **setRowIdAttrName**

PROCEDURE setRowIdAttrName(ctxHdl IN ctxType, attrName IN VARCHAR2)

Sets the name of the id attribute of the row enclosing tag. Passing null or an empty string for the tag results the row id attribute to be omitted.

#### **Parameters:**

ctxHdl - context handle

attrName - attribute name

### **setRowIdAttrValue**

PROCEDURE setRowIdAttrValue(ctxHdl IN ctxType, colName IN VARCHAR2)

Specifies the scalar column whose value is to be assigned to the id attribute of the row enclosing tag. Passing null or an empty string for the colName results the row id attribute being assigned the row count value (i.e. 0, 1, 2 and so on).

#### **Parameters:**

ctxHdl - context handle

colName - column whose value is to be assigned to the row id attr

### **setCollIdAttrName**

PROCEDURE setCollIdAttrName(ctxHdl IN ctxType, attrName IN VARCHAR2)

Sets the name of the id attribute of the collection element's separator tag. Passing null or an empty string for the tag results the row id attribute to be omitted.

#### **Parameters:**

ctxHdl - context handle

attrName - attribute name

### **useNullAttributeIndicator**

PROCEDURE useNullAttributeIndicator(ctxHdl IN ctxType, flag IN BOOLEAN)

Specified whether to use an XML attribute to indicate NULLness, or to do it by omitting the inclusion of the particular entity in the XML document.

#### **Parameters:**

ctxHdl - context handle

flag - use attribute to indicate null?

### **setTagCase**

PROCEDURE setTagCase(ctxHdl IN ctxType, tCase IN NUMBER)

Specified the case of the generated XML tags.

#### **Parameters:**

ctxHdl - context handle

tCase - the tag's case (0-asAre, 1-lower, 2-upper)

### **setDateFormat**

PROCEDURE setDateFormat(ctxHdl IN ctxType, mask IN VARCHAR2)

Sets the format of the generated dates in the XML doc. The syntax of the date format pattern (i.e. the date mask), should conform to the requirements of the java.text.SimpleDateFormat class. Setting the mask to null or an empty string, results the use of the default mask -- DEFAULT\_DATE\_FORMAT.

#### **Parameters:**

ctxHdl - context handle

mask - the date mask

### **setMaxRows**

PROCEDURE setMaxRows (ctxHdl IN ctxType, rows IN NUMBER)

Sets the max number of rows to be converted to XML. By default there is no max set.

#### **Parameters:**

ctxHdl - context handle

rows - max number of rows to generate

### **setSkipRows**

PROCEDURE setSkipRows(ctxHdl IN ctxType, rows IN NUMBER)

Sets the number of rows to skip. By default 0 rows are skipped.

#### **Parameters:**

ctxHdl - context handle

rows - number of rows to skip

**setStylesheetHeader**

PROCEDURE setStylesheetHeader(ctxHdl IN ctxType, uri IN VARCHAR2, type IN VARCHAR2 := 'text/xsl')

Sets the stylesheet header (i.e. stylesheet processing instructions) in the generated XML doc. Note: Passing null for the uri argument will unset the stylesheet header and the stylesheet type.

**Parameters:**

ctxHdl - context handle

uri - stylesheet URI

type - stylesheet type; defaults to 'text/xsl'

**setXSLT**

PROCEDURE setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null)

Registers a stylesheet to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a null for the uri argument.

**Parameters:**

ctxHdl - context handle

uri - stylesheet URI

ref - URL for include, import and external entities

**setXSLT**

PROCEDURE setXSLT(ctxHdl IN ctxType, stylesheet CLOB, ref IN VARCHAR2 := null)

Registers a stylesheet to be applied to generated XML. If a stylesheet was already registered, it gets replaced by the new one. To un-register the stylesheet pass in a null or an empty string for the stylesheet argument.

**Parameters:**

ctxHdl - context handle

stylesheet - the stylesheet

ref - URL for include, import and external entities

**setXSLTParam**

PROCEDURE setXSLTParam(ctxHdl IN ctxType, name IN VARCHAR2, value IN VARCHAR2)

Sets the value of a top-level stylesheet parameter. The parameter value is expected to be a valid XPath expression (note that string literal values would therefore have to be explicitly quoted). NOTE: if no stylesheet is registered, this method is a no op.

**Parameters:**

ctxHdl - context handle

name - name of the top level stylesheet parameter  
value - value to be assigned to the stylesheet parameter

### **removeXSLTParam**

PROCEDURE removeXSLTParam(ctxHdl IN ctxType, name IN VARCHAR2, value IN VARCHAR2)

Sets the value of a top-level stylesheet parameter. The parameter value is expected to be a valid XPath expression (note that string literal values would therefore have to be explicitly quoted). NOTE: if no stylesheet is registered, this method is a no op.

#### **Parameters:**

ctxHdl - context handle  
name - name of the top level stylesheet parameter

### **setBindValue**

PROCEDURE setBindValue(ctxHdl IN ctxType, bindName IN VARCHAR2, bindValue IN VARCHAR2)

Sets a value for a particular bind name.

#### **Parameters:**

ctxHdl - context handle  
bindName - bind name  
bindValue - bind value

### **setMetaHeader**

PROCEDURE setMetaHeader(ctxHdl IN ctxType, header IN CLOB := null)

Sets the XML meta header. When set, the header is inserted at the beginning of the metadata part (DTD or XMLSchema) of each XML document generated by this object. Note that the last meta header specified is the one that is used; furthermore, passing in null for the header, parameter unsets the meta header.

#### **Parameters:**

ctxHdl - context handle  
header - header

### **setDataHeader**

PROCEDURE setDataHeader(ctxHdl IN ctxType, header IN CLOB := null, tag IN VARCHAR2 := null)

Sets the xml data header. The data header is an XML entity which is appended at the beginning of the query-generated xml entity (ie. rowset). The two entities are enclosed by the tag specified via the docTag argument. Note that the last data header specified is the one that is used; furthermore, passing in null for the header, parameter unsets the data header.

**Parameters:**

ctxHdl - context handle

header - header

tag - tag used to enclose the data header and the rowset

**setRaiseException**

PROCEDURE setRaiseException(ctxHdl IN ctxType, flag IN BOOLEAN)

Tells the XSU to throw the raised exceptions. If this call isn't made or if false is passed to the flag argument, the XSU catches the SQL exceptions and generates an XML doc out of the exception's message.

**Parameters:**

ctxHdl - context handle

flag - throw raised exceptions?

**setRaiseNoRowsException**

PROCEDURE setRaiseNoRowsException(ctxHdl IN ctxType, flag IN BOOLEAN)

Tells the XSU to throw or not to throw an OracleXMLNoRowsException in the case when for one reason or another, the XML doc generated is empty. By default, the exception is not thrown.

**Parameters:**

ctxHdl - context handle

flag - throw OracleXMLNoRowsException if no data?

**propagateOriginalException**

PROCEDURE propagateOriginalException(ctxHdl IN ctxType, flag IN BOOLEAN)

Tells the XSU that if an exception is raised, and is being thrown, the XSU should throw the very exception raised; rather than, wrapping it with an OracleXMLSQLException.

**Parameters:**

ctxHdl - context handle

flag - propagate original exception?

**getExceptionContent**

PROCEDURE getExceptionContent(ctxHdl IN ctxType, errNo OUT NUMBER, errMsg OUT VARCHAR2)

Via its arguments, this method returns the thrown exception's error code and error message (i.e. sql error code) This is to get around the fact that the jvm throws an exception on top of whatever exception was raised; thus, rendering pl/sql unable to access the original exception.

**Parameters:**

ctxHdl - context handle  
errNo - error number  
errMsg - error message

### **getDTD**

FUNCTION getDTD(ctxHdl IN ctxType, withVer IN BOOLEAN := false) RETURN CLOB  
Generates the DTD based on the SQL query used to init. the context.

#### **Parameters:**

ctxHdl - context handle

withVer - generate the version info?

#### **Returns:**

The DTD.

### **getDTD**

PROCEDURE getDTD(ctx IN ctxType, xDoc IN CLOB, withVer IN BOOLEAN := false)  
Generates the DTD based on the SQL query used to init. the context.

#### **Parameters:**

ctxHdl - context handle

xDoc - lob into which to write the generated XML doc

withVer - generate the version info?

### **getXML**

FUNCTION getXML(sqlQuery IN VARCHAR2, metaType IN NUMBER := NONE) RETURN CLOB  
This is a convenience function. One doesn't have to explicitly open a context and close the context. This function creates the new context, executes the query, gets the xml back and closes the context..

#### **Parameters:**

sqlQueryl - SQL query

metaType - xml metadata type (i.e. NONE, DTD, SCHEMA)

#### **Returns:**

The XML document.

### **getXML**

FUNCTION getXML(sqlQuery IN CLOB, metaType IN NUMBER := NONE) RETURN CLOB

This is a convenience function. One doesn't have to explicitly open a context and close the context. This function creates the new context, executes the

query, gets the xml back and closes the context..

**Parameters:**

sqlQueryl - SQL query

metaType - xml metadata type (i.e. NONE, DTD, SCHEMA)

**Returns:**

The XML document.

**getXML**

FUNCTION getXML(ctxHdl IN ctxType, metaType IN NUMBER := NONE) RETURN CLOB

Generates the XML doc. based on the SQL query used to init. the context.

**Parameters:**

ctxHdl - context handle

metaType - xml metadata type (i.e. NONE, DTD, SCHEMA)

**Returns:**

The XML document.

**getXML**

PROCEDURE getXML(ctxHdl IN ctxType, xDoc IN CLOB, metaType IN NUMBER := NONE)

Generates the XML doc. based on the SQL query used to init. the context.

**Parameters:**

ctxHdl - context handle

xDoc - lob into which to write the generated XML doc

metaType - xml metadata type (i.e. NONE, DTD, SCHEMA)



# DBMS\_XMLSave

## Types:

### [ctxType](#)

The type of the query context handle. This the return type of "DBMS\_XMLSave.newContext()".

## Constants:

### [DEFAULT\\_ROWTAG](#)

The default tag name for the element cooresponding to db. records. -- ROW

### [DEFAULT\\_DATE\\_FORMAT](#)

Default date mask. -- 'MM/dd/yyyy HH:mm:ss'

### [MATCH\\_CASE](#)

Used to specify that when mapping XML elements to DB. entities the XSU should be case sensitive.

### [IGNORE\\_CASE](#)

Used to specify that when mapping XML elements to DB. entities the XSU should be case insensitive.

## Function and Procedure Index:

### [PROCEDURE clearKeyColumnList\(ctxType\)](#)

Clears the key column list.

### [PROCEDURE clearUpdateColumnList\(ctxType\)](#)

Clears the update column list.

### [PROCEDURE closeContext\(ctxType\)](#)

It closes/deallocates a particular save context

### [FUNCTION deleteXML\(ctxType, CLOB\) RETURN NUMBER](#)

Deletes records specified by data from the XML document, from the table specified at the context creation time.

### [FUNCTION deleteXML\(ctxType, VARCHAR2\) RETURN NUMBER](#)

Deletes records specified by data from the XML document, from the table specified at the context creation time.

**[PROCEDURE getExceptionContent\(ctxType, NUMBER, VARCHAR2\)](#)**

Via its arguments, this method returns the thrown exception's error code and error message (i.e.

**[FUNCTION insertXML\(ctxType, CLOB\) RETURN NUMBER](#)**

Inserts the XML document into the table specified at the context creation time.

**[FUNCTION insertXML\(ctxType, VARCHAR2\) RETURN NUMBER](#)**

Inserts the XML document into the table specified at the context creation time.

**[FUNCTION newContext\(targetTable IN VARCHAR2\) RETURN ctxType](#)**

It creates a save context, and it returns the context handle.

**[PROCEDURE propagateOriginalException\(ctxType, BOOLEAN\)](#)**

Tells the XSU that if an exception is raised, and is being thrown, the XSU should throw the very exception raised; rather than, wrapping it with an OracleXMLSQLException.

**[PROCEDURE setBatchSize\(ctxType, NUMBER\)](#)**

Changes the batch size used during DML operations.

**[PROCEDURE setCommitBatch\(ctxType, NUMBER\)](#)**

Sets the commit batch size.

**[PROCEDURE setDateFormat\(ctxType, VARCHAR2\)](#)**

Describes to the XSU the format of the dates in the XML document.

**[PROCEDURE setIgnoreCase\(ctxType, NUMBER\)](#)**

The XSU does mapping of XML elements to db.

**[PROCEDURE setKeyColumn\(ctxType, VARCHAR2\)](#)**

This methods adds a column to the "key column list".

**[PROCEDURE setRowTag\(ctxType, VARCHAR2\)](#)**

Names the tag used in the XML doc., to enclose the XML elements corresponding to db.

**[PROCEDURE setUpdateColumn\(ctxType, VARCHAR2\)](#)**

Adds a column to the "update column list".

**[PROCEDURE getExceptionContent\(ctxType, NUMBER, VARCHAR2\)](#)**

Updates the table specified at the context creation time with data from the XML document.

**[PROCEDURE propagateOriginalException\(ctxType, BOOLEAN\)](#)**

Updates the table specified at the context creation time with data from the

XML document.

## Functions and Procedures:

### **newContext**

FUNCTION newContext(targetTable IN VARCHAR2) RETURN ctxType

It creates a save context, and it returns the context handle.

**Parameters:**

targetTable - the target table into which to load the XML doc

**Returns:**

The context handle.

### **closeContext**

PROCEDURE closeContext(ctxHdl IN ctxType)

It closes/deallocates a particular save context

**Parameters:**

ctxHdl - context handle

### **setRowTag**

PROCEDURE setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2)

Names the tag used in the XML doc., to enclose the XML elements corresponding to db. records.

**Parameters:**

ctxHdl - context handle

tag - tag name

### **setIgnoreCase**

PROCEDURE setIgnoreCase(ctxHdl IN ctxType, flag IN NUMBER)

The XSU does mapping of XML elements to db. columns/attrs. based on the element names (xml tags). This function tells the XSU to do this match case insensitive.

**Parameters:**

ctxHdl - context handle

flag - ignore tag case in the XML doc? 0-false 1-true

### **setDateFormat**

PROCEDURE setDateFormat(ctxHdl IN ctxType, mask IN VARCHAR2)

Describes to the XSU the format of the dates in the XML document. The syntax of the date format pattern (i.e. the date mask), should conform to the requirements of the `java.text.SimpleDateFormat` class. Setting the mask to null or an empty string, results the use of the default mask -- `OracleXMLCore.DATE_FORMAT`.

**Parameters:**

ctxHdl - context handle

mask - the date mask

### **setBatchSize**

PROCEDURE setBatchSize(ctxHdl IN ctxType, batchSize IN NUMBER);

Changes the batch size used during DML operations. When performing inserts, updates or deletes, it is better to batch the operations so that they get executed in one shot rather than as separate statements. The flip side is that more memory is needed to buffer all the bind values. Note that when batching is used, a commit occurs only after a batch is executed. So if one of the statement inside a batch fails, the whole batch is rolled back. This is a small price to pay considering the performance gain; nevertheless, if this behaviour is unacceptable, then set the batch size to 1.

**Parameters:**

ctxHdl - context handle

batchSize - batch size

**See Also:**

[DEFAULT BATCH SIZE](#)

### **setCommitBatch**

PROCEDURE setCommitBatch(ctxHdl IN ctxType, batchSize IN NUMBER);

Sets the commit batch size. The commit batch size refers to the number or records inserted after which a commit should follow. Note that if `commitBatch` is `< 1` or the session is in "auto-commit" mode then the XSU does not make any explicit commit's. By default the commit-batch size is 0.

**Parameters:**

ctxHdl - context handle

batchSize - commit batch size

### **setUpdateColumn**

---

PROCEDURE setUpdateColumn(ctxHdl IN ctxType, colName IN VARCHAR2);

Adds a column to the "update column list". In case of insert, the default is to insert values to all the columns in the table; on the other hand, in case of updates, the default is to only update the columns corresponding to the tags present in the ROW element of the XML document. When the update column list is specified, the columns making up this list alone will get updated or inserted into.

**Parameters:**

ctxHdl - context handle

colName - column to be added to the update column list

**clearUpdateColumnList**

PROCEDURE clearUpdateColumnList(ctxHdl IN ctxType)

Clears the update column list.

**Parameters:**

ctxHdl - context handle

**See Also:**

[setUpdateColumn](#)

**setKeyColumn**

PROCEDURE setKeyColumn(ctxHdl IN ctxType, colName IN VARCHAR2)

This methods adds a column to the "key column list". In case of update or delete, it is the columns in the key column list that make up the where clause of the update/delete statement. The key columns list must be specified before updates can be done; yet, it is only optional for delete operations.

**Parameters:**

ctxHdl - context handle

colName - column to be added to the key column list

**clearKeyColumnList**

PROCEDURE clearKeyColumnList(ctxHdl IN ctxType)

Clears the key column list.

**Parameters:**

ctxHdl - context handle

**See Also:**

[setKeyColumn](#)**insertXML**

FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER

Inserts the XML document into the table specified at the context creation time.

**Parameters:**

ctxHdl - context handle

xDoc - string containing the XML document

**Returns:**

The number of rows inserted.

**insertXML**

FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER

Inserts the XML document into the table specified at the context creation time.

**Parameters:**

ctxHdl - context handle

xDoc - string containing the XML document

**Returns:**

The number of rows inserted.

**updateXML**

FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER

Updates the table specified at the context creation time with data from the XML document.

**Parameters:**

ctxHdl - context handle

xDoc - string containing the XML document

**Returns:**

The number of rows updated.

**updateXML**

FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER

Updates the table specified at the context creation time with data from the XML document.

**Parameters:**

ctxHdl - context handle

xDocl - string containing the XML document

**Returns:**

The number of rows updated.

**deleteXML**

FUNCTION deleteXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER

Deletes records specified by data from the XML document, from the table specified at the context creation time.

**Parameters:**

ctxHdl - context handle

xDoc - string containing the XML document

**Returns:**

The number of rows deleted.

**deleteXML**

FUNCTION deleteXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER

Deletes records specified by data from the XML document, from the table specified at the context creation time.

**Parameters:**

ctxHdl - context handle

xDocl - string containing the XML document

**Returns:**

The number of rows deleted.

**propagateOriginalException**

PROCEDURE propagateOriginalException(ctxHdl IN ctxType, flag IN BOOLEAN)

Tells the XSU that if an exception is raised, and is being thrown, the XSU should throw the very exception raised; rather than, wrapping it with an OracleXMLSQLException.

**Parameters:**

ctxHdl - context handle

flag - propagate original exception? 0-false 1-true

**getExceptionContent**

PROCEDURE getExceptionContent(ctxHdl IN ctxType, errNo OUT NUMBER, errMsg OUT VARCHAR2)

Via its arguments, this method returns the thrown exception's error code and error message (i.e. sql error code) This is to get around the fact that the jvm throws an exception on top of whatever exception was raised; thus, rendering pl/sql unable to access the original exception.

**Parameters:**

ctxHdl - context handle

errNo - error number

errMsg - error message



# Part VI

---

## XML Support

This section contains the following chapters:

- [Chapter 14, "XML Support"](#)



---

## XML Support

This chapter contains reference document on the types and functions related to XML support in the server including:

- [DBMS\\_XMLGEN](#) package
- [UriType](#) family
- [XMLType](#)

## DBMS\_XMLGEN

The DBMS\_XMLGEN is a package to convert the results of SQL queries to a canonical XML format. The package takes in any arbitrary SQL query and converts them into the XML format and returns the result as a CLOB.

The package is similar to the DBMS\_XMLQuery package, except that it is written in C and compiled into the kernel. This package may only be run on the database.

---

---

**Note:** DBMS\_XMLGEN is a built-in package in C. In general, use DBMS\_XMLGEN instead of DBMS\_XMLQuery wherever possible.

---

---

The package uses a canonical XML conversion. It does not support DTD or XMLSchema generation for now and these may be provided in later releases.

An example conversion is shown below:-

Assume an employee table with the following structure,

```
CREATE TABLE address_t AS OBJECT
(
    street VARCHAR2(20),
    state VARCHAR2(20),
    city VARCHAR2(20),
    zip VARCHAR2(20)
);
/
CREATE TABLE employee
(
    empno NUMBER,
    ename VARCHAR2(200),
    address address_t
);

insert into employee values (100,'John',
    address_t('100, Main Street','Jacksonville','FL','32607'));
insert into employee values (200,'Jack',
    address_t('200 Front Road','San Francisco','CA','94011'));

declare
    ctx dbms_xmlgen.ctxhandle;
    result clob;
begin
```

```

-- create a new context with the SQL query
ctx := dbms_xmlgen.newContext('select * from employee');

-- generate the CLOB as a result.
res := dbms_xmlgen.getXML(ctx);

-- print out the result of the CLOB
printClobOut(result); -- see the lob manual for examples on printing..
-- close the context
dbms_xmlgen.closeContext(ctx);
end;
/

```

produces a document like,

```

<?xml version="1.0"?>
<ROWSET>
  <ROW>
    <EMPNO>100</EMPNO>
    <ENAME>John</ENAME>
    <ADDRESS>
      <STREET>100 Main Street</STREET>
      <CITY>Jacksonville</CITY>
      <STATE>FL</STATE>
      <ZIP>32607</ZIP>
    </ADDRESS>
  </ROW>
  <ROW>
    <EMPNO>200</EMPNO>
    <ENAME>Jack</ENAME>
    <ADDRESS>
      <STREET>200 Front Street</STREET>
      <CITY>San Francisco</CITY>
      <STATE>CA</STATE>
      <ZIP>94011</ZIP>
    </ADDRESS>
  </ROW>
</ROWSET>

```

As you can see from the example, each row of the result gets translated to a ROW element which contains all the columns. The columns themselves are converted to nested elements with the column name as the element tag name. The object columns (such as address) retain their structure with the object type attributes becoming nested elements of the column.

Thus, deep structuring of the XML can be achieved easily using the object-relational views and tables.

## DBMS\_XMLGEN Procedures and Functions

The DBMS\_XMLGEN package contains a variety of functions to generate XML and to set the properties of the result.

The order of calling procedures are as follows

1. Call `newContext` to get a new context handle given the SQL query.
2. Call `setRowTag` or `setRowSetTag` to change the names of tags.
3. Set the `maxRows` and `skipRows` if needed to change the number of rows fetched or skipped.
4. Call `getXML` to get the XML value. You can call this repeatedly to obtain XML from the next set of rows.
5. You can check if any rows were processed using the `getNumRowsProcessed` method.
6. Call `closeContext` to close the context handle and release resources.

### FUNCTION `newContext(varchar2) RETURN ctxHandle`

Creates a new context handle from a passed in SQL query. The context handle can be used for the rest of the functions.

### PROCEDURE `setRowTag(ctxHandle, varchar2);`

Sets the name of the element enclosing each row of the result. The default tag is "ROW"

### PROCEDURE `setRowSetTag(ctxHandle, varchar2);`

Sets the name of the element enclosing the entire result. The default tag is "ROWSET"

### PROCEDURE `getXML(ctxHandle, clob, number);`

This appends the XML to the CLOB passed in. Use the `getNumRowsProcessed` function to figure out if any rows were appended.

### FUNCTION `getXML(ctxHandle, number) RETURN CLOB;`

This returns the XML as a CLOB.

**FUNCTION getXMLType(ctxHandle, number) RETURN sys.XMLType;**

This returns the XML as an XMLType.

**FUNCTION getNumRowsProcessed(ctxHandle) RETURN number;**

This gets the number of SQL rows that were processed in the last call to getXML. You can call this to find out if the end of the result set has been reached.

**PROCEDURE setMaxRows(ctxHandle, number);**

This sets the maximum number of rows to be fetched each time.

**PROCEDURE setSkipRows(ctxHandle, number);**

This sets the number of rows to skip everytime before generating the XML. The default is 0.

**PROCEDURE setConvertSpecialChars(ctxHandle, boolean);**

This sets whether special characters such as \$ etc.. which are non-XML characters should be converted or not to their escaped representation. The default is to perform the conversion.

**PROCEDURE useItemTagsForColl(ctxHandle);**

This forces the use of the collection column name appended with the tag "\_ITEM" for collection elements. The default is to set the underlying object type name for the base element of the collection.

**PROCEDURE restartQuery(ctx IN ctxHandle);**

Restart the query to start fetching from the beginning.

**PROCEDURE closeContext(ctx IN ctxHandle);**

Close the context and release all resources.

## DBMS\_XMLGEN Type definitons

SUBTYPE **ctxHandle** IS NUMBER

This is the context handle that is used by all functions.

### DTD or schema specifications

- NONE CONSTANT NUMBER := 0; -- supported for this release.
- DTD CONSTANT NUMBER := 1;
- SCHEMA CONSTANT NUMBER := 2;

These constants may be used in the `getXML` function to specify whether to generate a DTD or a schema or none. Note that only the NONE specification is supported in the `getXML` functions for this release.

## Functions Prototypes

### `newContext()`

#### PURPOSE

Given a query string, generate a new context handle to be used in subsequent functions.

#### SYNTAX

```
FUNCTION newContext(queryString IN VARCHAR2) RETURN ctxHandle;
```

#### PARAMETERS

`queryString` (IN)- the query string, the result of which needs to be converted to XML

#### RETURNS

The context handle

#### COMMENTS

You need to call this function first to obtain a handle that you can use in the `getXML()` and other functions to get the XML back from the result.



## setRowTag()

### PURPOSE

Set the name of the element separating all the rows. The default name is ROW.

### SYNTAX

```
PROCEDURE setRowTag(ctx IN ctxHandle, rowTag IN VARCHAR2);
```

### PARAMETERS

ctx (IN) - the context handle obtained from the newContext call

rowTag (IN) - the name of the ROW element.

NULL indicates that you do not want the ROW element to be present.

### COMMENTS

You can call this function to set the name of the ROW element, if you do not want the default "ROW" name to show up. You can also set this to NULL to suppress the ROW element itself. However, this is an error if both the row and the rowset are null and there is more than one column or row in the output.

## setRowSetTag()

### PURPOSE

Set the name of the document's root element. The default name is "ROWSET"

### SYNTAX

```
PROCEDURE setRowSetTag(ctx IN ctxHandle, rowSetTag IN VARCHAR2);
```

### PARAMETERS

ctx (IN) - the context handle obtained from the newContext call

rowsetTag (IN) - the name of the document element.

NULL indicates that you do not want the ROW element to be present.

### COMMENTS

You can call this function to set the name of the document root element, if you do not want the default "ROWSET" name in the output. You can also set this to NULL to suppress the printing of this element. However, this is an error if both the row and the rowset are null and there is more than one column or row in the output.

## getXML()

### PURPOSE

Get the XML document by fetching the maximum number of rows specified. It appends the XML document to the CLOB passed in.

### SYNTAX

```
PROCEDURE getXML(ctx IN ctxHandle, clobval IN OUT NCOPY clob,  
dtdOrSchema IN number := NONE)
```

### PARAMETERS

ctx (IN) - The context handle obtained from the newContext() call.  
clobval (IN/OUT) - the clob to which the XML document is to be appended.  
dtdOrSchema (IN) - whether we should generate the Dtd or Schema or neither (NONE). NONE is the only option currently supported.

### COMMENTS

Use this version of the getXML function, if you want to avoid any extra CLOB copies and you want to reuse the same CLOB for subsequent calls. The user must create the lob locator, but this is a one-time cost, and so it is recommended that users use this procedure whenever possible.

When generating the XML, the number of rows indicated by the setSkipRows call are skipped, then the maximum number of rows as specified by the setMaxRows call (or the entire result if not specified) is fetched and converted to XML.

Use the getNumRowsProcessed function to check if any rows were retrieved or not.

## getXML()

### PURPOSE

Generate the XML document and return that as a CLOB.

### SYNTAX

```
FUNCTION getXML(ctx IN ctxHandle, dtdOrSchema IN number := NONE) RETURN clob
```

### PARAMETERS

ctx (IN) - The context handle obtained from the newContext() call.  
dtdOrSchema (IN) - whether we should generate the Dtd or Schema or neither (NONE). NONE is the only option currently supported.

**RETURNS**

A temporary CLOB containing the document.

**COMMENTS**

You need to free the temporary CLOB that is obtained from this function using the `dbms_lob.freetemporary` call.

**getXMLType()****PURPOSE**

Generate the XML document and return it as `sys.XMLType`.

**SYNTAX**

```
FUNCTION getXMLType(ctx IN ctxHandle, dtdOrSchema IN number := NONE)  
RETURN sys.XMLType
```

**PARAMETERS**

`ctx` (IN) - The context handle obtained from the `newContext()` call.  
`dtdOrSchema` (IN) - whether we should generate the Dtd or Schema or neither (NONE). NONE is the only option supported currently.

**RETURNS**

XML document as an `XMLType`

**COMMENTS**

Further `XMLType` operations can be done on the result, such as `ExistsNode` and `Extract`. This also provides a way to get the result back as string using `getStringVal()`, if the result size is known to be less than 4K.

**getNumRowsProcessed()****PURPOSE**

Get the number of SQL rows processed when generating the XML using the `getXML` call. This count does not include the number of rows skipped before generating the XML.

## SYNTAX

```
FUNCTION getNumRowsProcessed(ctx IN ctxHandle) RETURN number
```

## PARAMETERS

queryString (IN)- the query string, the result of which needs to be converted to XML

## RETURNS

The number of rows processed in the last call to getXML. This does not include the number of rows skipped.

## COMMENTS

Use this function to determine the terminating condition if you are calling getXML in a loop. Note that getXML would always generate a XML document even if there are no rows present.

## setMaxRows()

### PURPOSE

Set the maximum number of rows to fetch from the SQL query result for every invocation of the getXML call.

### SYNTAX

```
PROCEDURE setMaxRows(ctx IN ctxHandle, maxRows IN NUMBER);
```

### PARAMETERS

ctx (IN) - the context handle corresponding to the query executed  
maxRows (IN) - the maximum number of rows to get per call to getXML

### COMMENTS

The maxRows paramter can be used when generating paginated results using this utility. For instance when generating a *page* of XML or HTML data, you can restrict the number of rows converted to XML and then in subsequent calls, you can get the next set of rows and so on.

This also can provide for faster reponse times.

## setSkipRows()

### PURPOSE

Skip a given number of rows before generating the XML output for every call to the `getXML` routine.

### SYNTAX

```
PROCEDURE setSkipRows(ctx IN ctxHandle, skipRows IN NUMBER);
```

### PARAMETERS

`ctx` (IN) - the context handle corresponding to the query executed

`skipRows` (IN) - the number of rows to skip per call to `getXML`

### COMMENTS

The `skipRows` paramter can be used when generating paginated results for stateless web pages using this utility. For instance when generating the first *page* of XML or HTML data, you can set `skipRows` to zero. For the next set, you can set the `skipRows` to the number of rows that you got in the first case.

## setConvertSpecialChars()

### PURPOSE

Set whether special characters in the XML data need to be converted into their escaped XML equivalent or not. For example, the "<" sign is converted to `&lt;`. The default is to perform conversions.

### SYNTAX

```
PROCEDURE setConvertSpecialChars(ctx IN ctxHandle, conv IN boolean);
```

### PARAMETERS

`ctx` (IN) - the context handle to use

`conv` (IN) - true indicates that conversion is needed.

### COMMENTS

You can use this function to speed up the XML processing whenever you are sure that the input data cannot contain any special characters such as `<`, `>`, `"`, `'` etc. which need to be escaped. Note that it is expensive to actually scan the character data to replace the special characters, particularly if it involves a lot of data. So in cases

when the data is XML-safe, then this function can be called to improve performance.

## **useItemTagsForColl()**

### **PURPOSE**

Set the name of the collection elements. The default name for collection elements is the type name itself. You can override that to use the name of the column with the "\_ITEM" tag appended to it using this function.

### **SYNTAX**

```
PROCEDURE useItemTagsForColl(ctx IN ctxHandle);
```

### **PARAMETERS**

ctx (IN) - the context handle

### **COMMENTS**

If you have a collection of NUMBER, say, the default tag name for the collection elements is NUMBER. You can override this behavior and generate the collection column name with the \_ITEM tag appended to it, by calling this procedure.

## **restartQuery()**

### **PURPOSE**

Restart the query and generate the XML from the first row again.

### **SYNTAX**

```
PROCEDURE restartQuery(ctx IN ctxHandle);
```

### **PARAMETERS**

ctx (IN) - the context handle corresponding to the current query

### **COMMENTS**

You can call this to start executing the query again, without having to create a new context.

## closeContext()

### PURPOSE

Closes a given context and releases all resources associated with that context, including the SQL cursor and bind and define buffers etc.

### SYNTAX

```
PROCEDURE closeContext(ctx IN ctxHandle);
```

### PARAMETERS

ctx (IN) - the context handle to close

### COMMENTS

Closes all resources associated with this handle. After this you cannot use the handle for any other DBMS\_XMLGEN function call.

## URI Support

Oracle9i supports the UriType family of types which can be used to store and query Uri-refs inside the database. The UriType itself is an abstract object type and the HttpUriType and DBUriType are subtypes of it.

You can create a UriType column and store instances of the DBUriType or the HttpUriType inside of it.

You can also define your own subtypes of the UriType to handle different URL protocols.

Oracle9i also provides a UriFactory package which can be used as a factory method to generate various instances of these UriTypes automatically by scanning the prefix (e.g. http:// or ftp:// etc..). You can also register your subtype and give the prefix that you support. For instance if you have written a subtype to handle the gopher protocol, you can register that the prefix "gopher://" be handled by your subtype. With that, the UriFactory will generate your subtype instance for any URL starting with that prefix.

## UriType

The UriType is the abstract super type. It provides a standard set of functions to get the value pointed to by the Uri. The actual implementation of the protocol must be defined by the subtypes of this type.

You cannot create instances of this type directly. However, you can create columns of this type and store subtype instances in it.

For example,

```
create table uri_tab ( url sys.uritype);

insert into uri_tab values
  (sys.httpuritype.createHttpuri('http://www.oracle.com'));
insert into uri_tab values (
  sys.dburitype.createdburi('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]'));

-- Now you can select from the column without having to know
-- what instance of the URL is actually stored.
select e.url.getclob() from uri_tab e;

-- would retrieve both the HTTP URL and the DBUri-ref.
```

## Member functions

### MEMBER FUNCTION `getClob()` RETURN `clob`

This member function returns a CLOB by following the URL and getting the document.

### MEMBER FUNCTION `getExternalUri()` RETURN `varchar2`

Get the url stored in the *external* form, i.e. the non-Url characters such as spaces are escaped, using the standard escape sequence, %xx where xx is the hexadecimal value of the UTF-8 encoding of the character.

### MEMBER FUNCTION `getUri()` RETURN `varchar2`;

Returns the URL stored in the type without any escaping. This function must be used instead of directly referencing the url attribute in the type.



## Function prototypes

### getClob()

#### **PURPOSE**

Get the document pointed to by the URL as a CLOB. This function can be overridden in the subtype instances.

#### **SYNTAX**

```
MEMBER FUNCTION getClob() RETURN clob
```

#### **RETURNS**

A temporary or permanent lob containing the document pointed to by the URL

#### **COMMENTS**

You need to free the clob if it is a temporary CLOB.

### getExternalUrl()

#### **PURPOSE**

Get the URL stored inside the instance after escaping the non-URL characters as specified in the Uri-ref specification.

#### **SYNTAX**

```
MEMBER FUNCTION getExternalUrl() RETURN varchar2
```

#### **COMMENTS**

Generates the escaped URL that can be used in HTML web pages. The subtype instances override this member function to provide additional semantics. For instance, the `HttpUriType` does not store the prefix `http://` in the `Url` itself. When generating the external `Url`, it appends the prefix and generates it. This is the reason that you should use the `getExternalUrl` or `getUrl` to get to the URL value instead of using the attribute present in the `UriType`.

## getUrl()

### PURPOSE

Get the URL stored inside the instance without escaping the non-URL characters.

### SYNTAX

MEMBER FUNCTION getUrl() RETURN varchar2

### COMMENTS

Generates the URL that is present in the UriType instance. The subtype instances override this member function to provide additional semantics. For instance, the HttpUriType does not store the prefix `http://` in the Url itself. When generating the external Url, it appends the prefix and generates it. This is the reason that you should use the `getExternalUrl` or `getUrl` to get to the URL value instead of using the attribute present in the UriType.

## HttpUriType

The `HttpUriType` is a subtype of the `UriType` that provides support for the HTTP protocol. This uses the `UTL_HTTP` package underneath to access the HTTP urls. This does not support the proxy or secure wallets for this release.

For example,

```
-- create a uri table to store the Http instances
create table uri_tab ( url sys.httppuritype);

-- insert the Http instance.
insert into uri_tab values
    (sys.httppuritype.createUri('http://www.oracle.com'));

-- generate the HTML
select e.url.getclob() from uri_tab e;
```

## Member functions

### MEMBER FUNCTION getClob() RETURN clob

This member function returns a CLOB by following the HTTPURL and getting the document.

**MEMBER FUNCTION `getExternalUrl()` RETURN `varchar2`**

Get the HTTP url stored in the *external* form, i.e. the non-Url characters such as spaces are escaped, using the standard escape sequence, %xx where xx is the hexadecimal value of the UTF-8 encoding of the character.

**MEMBER FUNCTION `getUrl()` RETURN `varchar2`;**

Returns the URL stored in the type without any escaping. This function must be used instead of directly referencing the url attribute in the type.

**STATIC FUNCTION `createUrl(url IN varchar2)` RETURN `HttpUriType`**

Static function to construct a `HttpUri` instance.

## Function prototypes

**`getClob()`****PURPOSE**

Get the document pointed to by the HTTP URL as a CLOB.

**SYNTAX**

```
MEMBER FUNCTION getClob() RETURN clob
```

**RETURNS**

A temporary lob containing the document pointed to by the URL

**COMMENTS**

You need to free the temporary clob.

**`getExternalUrl()`****PURPOSE**

Get the URL stored inside the instance after escaping the non-URL characters as specified in the Uri-ref specification.

## SYNTAX

MEMBER FUNCTION `getExternalUrl()` RETURN `varchar2`

## COMMENTS

Generates the escaped URL that can be used in HTML web pages. The subtype instances override this member function to provide additional semantics. The `HttpUriType` does not store the prefix `http://` in the `Url` itself. When generating the external `Url`, it appends the prefix and generates it.

## `getUrl()`

### PURPOSE

Get the URL stored inside the instance without escaping the non-URL characters.

### SYNTAX

MEMBER FUNCTION `getUrl()` RETURN `varchar2`

### COMMENTS

Generates the URL that is present in the `HttpUri` instance.

## `createUrl()`

### PURPOSE

Static function to construct a `Http URL` instance.

### SYNTAX

STATIC FUNCTION `createUrl(url IN varchar2)` RETURN `HttpUriType`

### PARAMETERS

`url (IN)` - the `url` string containing a valid HTTP URL.

The `url` string is assumed to be in the escaped form. i.e. non-`url` characters are represented as `%xx` where `xx` is the hexadecimal value for the UTF-8 encoding of the character.

### COMMENTS

Parses the `Http URL` supplied and generates a `HttpUri` instance. Note that this instance does not contain the prefix `"http://"` in the `url` stored.

## DbUriType

The DbUriType is a subtype of the UriType that provides support for of DBUri-refs. A DBUri-ref is a intra-database URL that can be used to reference any row or row-column data in the database.

The URL is specified as an XPath expression over a XML visualization of the database. The schemas become elements which contain tables and views. These tables and view further contain the rows and columns inside them.

For example, the virtual document that a user scott can see can be something like this,

```
<?xml version="1.0"?>
<DATABASE>
  <SCOTT>
    <EMPLOYEE>
      <ROWSET>
        <ROW>
          <EMPNO>100</EMPNO>
          <ENAME>John</ENAME>
          <ADDRESS>
            <STREET>100 Main Street</STREET>
            <CITY>Jacksonville</CITY>
            <STATE>FL</STATE>
            <ZIP>32607</ZIP>
          </ADDRESS>
        </ROW>
        <ROW>
          <EMPNO>200</EMPNO>
          <ENAME>Jack</ENAME>
          <ADDRESS>
            <STREET>200 Front Street</STREET>
            <CITY>San Francisco</CITY>
            <STATE>CA</STATE>
            <ZIP>94011</ZIP>
          </ADDRESS>
        </ROW>
      </ROWSET>
    </EMPLOYEE>
  </SCOTT>
</DATABASE>
```

Hence to reference the State attribute inside the employee table, you can formulate a DBUri-ref as shown below:-

```
/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]/ADDRESS/STATE
```

You can use the DBUriType to create instances of these and store them in columns.

```
-- create a table
create table dburi_tab (dburl sys.dburitype);

-- insert values..!
insert into dburi_tab values (
  sys.dburitype.createdUri('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]/ADDRESS/STATE'));

select e.dburl.getclob() from dburi_tab e;

-- will return,
<?xml version="1.0"?>
<STATE>CA</STATE>
```

You can also generate the DBUri-ref dynamically using the SYS\_DBURIGEN SQL function.

For example you can generate a DBUri-ref to the state attribute as shown below:-

```
select sys_dburigen(e.ename,e.address.state) AS urlcol
from scott.employee e;
```

## Member functions

### MEMBER FUNCTION getClob() RETURN clob

This member function returns a CLOB by following the DBUriType and getting the document.

### MEMBER FUNCTION getExternalUrl() RETURN varchar2

Get the DbUri-ref stored in the *external* form, i.e. the non-Url characters such as spaces are escaped, using the standard escape sequence, %xx where xx is the hexadecimal value of the UTF-8 encoding of the character.

### MEMBER FUNCTION getUrl() RETURN varchar2;

Returns the URL stored in the type without any escaping. This function must be used instead of directly referencing the url attribute in the type.

### STATIC FUNCTION createUri(url IN varchar2) RETURN DbUriType

Static function to construct a DbUri instance.

## Function prototypes

### getClob()

#### **PURPOSE**

Get the document pointed to by the DBUri-ref

#### **SYNTAX**

MEMBER FUNCTION getClob() RETURN clob

#### **RETURNS**

A temporary lob containing the document pointed to by the URL

#### **COMMENTS**

You need to free the temporary clob.

The document obtained may be a XML document or a text document. When the DBUri-ref identifies an element in the XPath, the result is a well-formed XML document. On the other hand, if it identifies a text node (using the text() function), then we get back only the text contents of the column or attribute.

### getExternalUrl()

#### **PURPOSE**

Get the URL stored inside the instance after escaping the non-URL characters as specified in the Uri-ref specification.

#### **SYNTAX**

MEMBER FUNCTION getExternalUrl() RETURN varchar2

#### **COMMENTS**

Generates the escaped URL that can be used in HTML web pages. You would still have to append the DBUri servlet that can process the DBUri-ref before using it in web pages.

## getUrl()

### PURPOSE

Get the URL stored inside the instance without escaping the non-URL characters.

### SYNTAX

```
MEMBER FUNCTION getUrl() RETURN varchar2
```

### COMMENTS

Generates the URL that is present in the HttpUri instance.

## createUri()

### PURPOSE

Static function to construct a DbUri-ref instance.

### SYNTAX

```
STATIC FUNCTION createUri(url IN varchar2) RETURN DbUriType
```

### PARAMETERS

url (IN) - the url string containing a valid HTTP URL.

The url string is assumed to be in the escaped form. i.e. non-url characters are represented as %xx where xx is the hexadecimal value for the UTF-8 encoding of the character.

### COMMENTS

Parses the URL given and creates a Uri-ref type instance.

## UriFactory Package

The UriFactory package contains factory methods that can be used to generate the appropriate instance of the Uri types without having to hard code the implementation in the program.

The UriFactory package also provides the ability to register new subtypes of the UriType to handle various other protocols not supported by Oracle9i. For example, one can invent a new protocol "ecom://" and define a subtype of the UriType to handle that protocol and register it with UriFactory. After that any factory method would generate the new subtype instance if it sees the ecom prefix.



For example,

```

create table url_tab (urlcol varchar2(20));

-- insert a Http reference
insert into url_tab values ('http://www.oracle.com');

-- insert a DBUri-ref reference
insert into url_tab values ('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]');

-- create a new type to handle a new protocol called ecom://
create type EComUriType under UriType
(
  overriding member function getClob() return clob,
  overriding member function getBlob() return blob, -- not supported
  overriding member function getExternalUrl() return varchar2,
  overriding member function getUrl() return varchar2,

  -- MUST NEED THIS for registering with the url handler
  static member function createUri(url in varchar2) return EComUriType
);
/

-- register a new protocol handler.
begin

  -- register a new handler for ecom:// prefixes. The handler
  -- type name is ECOMURITYPE, schema is SCOTT
  -- Ignore the prefix case, when comparing and also strip the prefix
  -- before calling the createUri function
  urifactory.registerHandler('ecom://', 'SCOTT', 'ECOMURITYPE',
    true,true);
end;
/
insert into url_tab values ('ECOM://company1/company2=22/comp');

-- now use the factory to generate the instances.!
select urifactory.getUri(urlcol) from url_tab;

-- would now generate
HttpUriType('www.oracle.com'); -- a Http uri type instance

DBUriType('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"],null); -- a DBUriType

EComUriType('company1/company2=22/comp'); -- a EComUriType instance

```

## Package functions

### FUNCTION `getUri(vvarchar2)` RETURN `UriType`

This function takes in the Url string and generates the appropriate handler instance.

### FUNCTION `escapeUri(vvarchar2)` RETURN `vvarchar2`

Escape the given URL in to the *external* form, i.e. the non-Url characters such as spaces are escaped, using the standard escape sequence, %xx where xx is the hexadecimal value of the UTF-8 encoding of the character.

### FUNCTION `unescapeUri(vvarchar2)` RETURN `vvarchar2`;

Un-escape a URL. i.e. convert the %xx escape sequence into the corresponding characters.

### PROCEDURE `registerUrlHandler(vvarchar2, vvarchar2, vvarchar2, boolean, boolean)`

This procedure registers a Url handler given the prefix to scan for and the schema and type name that handles the given URL.

### PROCEDURE `unRegisterUrlHandler(vvarchar2)`

This function deletes a handler for a particular prefix.

## Function prototypes

### `getUri()`

#### **PURPOSE**

Factory method to get the correct Url handler for the given URL string. Returns a subtype instance of the `UriType`.

#### **SYNTAX**

```
FUNCTION getUri(url IN Vvarchar2) RETURN UriType
```

## PARAMETERS

`url` (IN) - the `url` string containing a valid HTTP URL.

The `url` string is assumed to be in the escaped form. i.e. non-`url` characters are represented as `%xx` where `xx` is the hexadecimal value for the UTF-8 encoding of the character.

## RETURNS

A subtype instance of the `UriType` which can handle the protocol. By default it always creates a `DBuri-ref` instance, if it cannot resolve the URL.

## COMMENTS

The user can register a URL handler for a particular prefix using the `registerUrlHandler` and the `getUrl` would use that subtype if the prefix matches.

## `escapeUri()`

### PURPOSE

Escape the `url` string by replacing the non-URL characters as specified in the `Uri-ref` specification by their equivalent escape sequence.

### SYNTAX

MEMBER FUNCTION `escapeUri()` RETURN `varchar2`

### PARAMETERS

`url` (IN) - the `url` string containing a valid URL.

### COMMENTS

Generates the escaped URL that can be used in HTML web pages. The subtype instances override this member function to provide additional semantics. For instance, the `HttpUriType` does not store the prefix `http://` in the `Url` itself. When generating the external `Url`, it appends the prefix and generates it. This is the reason that you should use the `getExternalUrl` or `getUrl` to get to the URL value instead of using the attribute present in the `UriType`.

## unescapeUri()

### PURPOSE

Un-escape a given url.

### SYNTAX

```
FUNCTION unescapeUri() RETURN varchar2
```

### PARAMETERS

url (IN) - the url string that needs to be un-escaped

### COMMENTS

Reverse of the escapeUri. Scans the string and converts any %xx into the equivalent UTF-8 character. Since the return type is a VARCHAR2, the character would be converted into the equivalent character as defined by the database character set.

## registerUrlHandler()

### PURPOSE

Register a particular type name for handling a particular URL.

### SYNTAX

```
PROCEDURE registerUrlHandler(prefix IN varchar2, schemaName in varchar2,  
    typename in varchar2, ignoreCase in boolean := true,  
    stripprefix in boolean := true)
```

### PARAMETERS

prefix (IN) - the prefix to handle (e.g. http://)

schemaName (IN) - the name of the schema in which the type resides

typename (IN) - the name of the type to handle the URL

Both the typename and schemaname are case sensitive.

ignorecase (IN) - should the case be ignored when matching prefixes?

stripprefix (IN) - should the prefix be stripped before generating the instance of the type?

**COMMENTS**

Register a URL handler. The type specified must be valid and **MUST** be a subtype of the UriType or one of its subtypes. The type **MUST** also implement the following static member function

```
STATIC FUNCTION createUri(url IN varchar2) RETURN <typename>;
```

This function is called by the getUrl() to generate an instance of the type. The stripprefix indicates that the prefix must be stripped off before calling this function.

**unRegisterUrlHandler()****PURPOSE**

Un-register a Url handler.

**SYNTAX**

```
PROCEDURE unregisterUrlHandler(prefix in varchar2)
```

**PARAMETERS**

prefix (IN) - the prefix that should be unregistered.

**COMMENTS**

Un-registers a particular prefix. Note that this would only un-register user registered handler prefixes and not the sytem predefined prefixes such as http://

## XMLType

Oracle9i has introduced a new datatype for handling XML data. This datatype is a system defined object type that has predefined member function on it to extract XML nodes and fragments.

You can create columns of XMLType and insert XML documents into it. You can also generate XML documents as XMLType instances dynamically using the SYS\_XMLGEN and SYS\_XMLAGG SQL functions.

For example,

```
create table xml_tab ( xmlval sys.xmltype);

insert into xml_tab values (
    sys.xmltype.createxml('<?xml version="1.0"?>
        <EMP>
            <EMPNO>221</EMPNO>
            <ENAME>John</ENAME>
        </EMP>' ));

insert into xml_tab values (
    sys.xmltype.createxml('<?xml version="1.0"?>
        <PO>
            <PONO>331</PONO>
            <PONAME>PO_1</PONAME>
        </PO>' ));

-- now extract the numerical values for the employee numbers
select e.xmlval.extract('//EMPNO/text()').getNumVal() as empno
from xml_tab
where e.xmlval.existsnode('/EMP/EMPNO') = 1;
```

## Member functions

### STATIC FUNCTION createXML(varchar2) RETURN sys.XMLType

This static function creates an XMLType instance from a string.

### STATIC FUNCTION createXML(clob) RETURN sys.XMLType

This static function creates an XMLType instance from a CLOB.

**MEMBER FUNCTION existsNode(varchar2) RETURN number**

Checks if the given XPath expression returns any result nodes. Returns 1 for true.

**MEMBER FUNCTION extract(varchar2) RETURN sys.XMLType**

Applies the XPath expression over the XML data to return a XMLType instance containing the resultant fragment.

**MEMBER FUNCTION isFragment() RETURN number**

Returns 1 or 0 indicating if the XMLType instance contains a fragment or a well-formed document.

**MEMBER FUNCTION getClobVal() RETURN clob**

Returns the document as a CLOB.

**MEMBER FUNCTION getNumVal() RETURN clob**

Returns the fragment or text value in the XMLType to a number.

**MEMBER FUNCTION getStringVal() RETURN varchar2**

Returns the document as a string.

## Function prototypes

**createXML()****PURPOSE**

Static function to create the XMLType instance from a string.

**SYNTAX**

```
STATIC FUNCTION createXML(xmlval IN varchar2) RETURN sys.XMLType deterministic
```

**PARAMETERS**

xmlval (IN) - string containing the XML document

### **RETURNS**

An XMLType instance.

### **COMMENTS**

The string must contain a well-formed and valid XML document.

## **createXML()**

### **PURPOSE**

Static function to create the XMLType instance from a CLOB.

### **SYNTAX**

```
STATIC FUNCTION createXML(xmlval IN clob) RETURN sys.XMLType deterministic
```

### **PARAMETERS**

xmlval (IN) - CLOB containing the XML document

### **RETURNS**

An XMLType instance.

### **COMMENTS**

The CLOB must contain a well-formed and valid XML document.

## **existsNode()**

### **PURPOSE**

Given an XPath expression, checks if the XPath applied over the document can return any valid nodes.

### **SYNTAX**

```
MEMBER FUNCTION existsNode(xpath IN varchar2) RETURN number deterministic
```

### **PARAMETERS**

xpath (IN) - the XPath expression to test



**RETURNS**

0 if the XPath expression does not return any nodes else 1.

**COMMENTS**

If the XPath string is null or the document is empty, then a value of 0 is returned.

**extract()****PURPOSE**

Given an XPath expression, applies the XPath to the document and returns the fragment as an XMLType

**SYNTAX**

MEMBER FUNCTION extract(xpath IN varchar2) RETURN sys.XMLType deterministic

**PARAMETERS**

xpath (IN) - the XPath expression to apply

**RETURNS**

An XMLType instance containing the result node(s).

**COMMENTS**

If the XPath does not result in any nodes, then the result is NULL.

**isFragment()****PURPOSE**

Determine if the XMLType instance corresponds to a well-formed document, or a fragment.

**SYNTAX**

MEMBER FUNCTION isFragment() RETURN boolean deterministic

**RETURNS**

Returns 1 or 0 indicating if the XMLType instance contains a fragment or a well-formed document.

## **getClobVal()**

### **PURPOSE**

Gets the document as a CLOB

### **SYNTAX**

MEMBER FUNCTION getClobVal() RETURN clob deterministic

### **RETURNS**

An CLOB containing the seralized XML representation.

### **COMMENTS**

You need to free the temporary CLOB after use.

## **getNumVal()**

### **PURPOSE**

Gets the numeric value pointed to by the XMLType as a number

### **SYNTAX**

MEMBER FUNCTION getNumVal() RETURN number deterministic

### **RETURNS**

An number formatted from the text value pointed to by the XMLType instance.

### **COMMENTS**

The XMLType must point to a valid text node that contains a numerical value.

## **getStringVal()**

### **PURPOSE**

Gets the document as a string.

### **SYNTAX**

MEMBER FUNCTION getStringVal() RETURN varchar2 deterministic

**RETURNS**

A string containing the seralized XML representation, or in case of text nodes, the text itself.

**COMMENTS**

If the XML document is bigger than the maximum size of the varchar2, which is 4000, then an error is raised at run time.



---

---

# Index

## A

---

activeFound(), 4-23  
addAttribute(String, String), 2-9  
addCDATASection(String), 2-5  
addData(String), 2-5  
addDOMBuilderErrorListener(DOMBuilderErrorListener), 4-4  
addDOMBuilderListener(DOMBuilderListener), 4-5  
addElement(Object), 2-9  
addNode(CGNode), 2-5  
addXSLTransformerErrorListener(XSLTransformerErrorListener), 4-26  
addXSLTransformerListener(XSLTransformerListener), 4-26  
ANY, 1-32  
appendChild(Node), 1-107  
ASTERISK, 1-32  
AttListDecl, 1-136  
AttName, 1-137  
ATTRDECL, 1-107  
Attribute, 1-137  
AttValue, 1-137

## C

---

CANTREAD\_XSQL, 3-6  
CANTREAD\_XSQL\_MSG, 3-6  
CDATA, 1-6  
cDATASection(char[], int, int), 1-11, 1-87  
CDSect, 1-137  
CGDocument, 2-2  
CGDocument(), 2-2

CGDocument(String, DTD), 2-2  
CGNode, 2-4  
CGNode(), 2-4  
CGNode(String), 2-4  
CGXSDElement, 2-9  
CGXSDElement(), 2-9  
CharData, 1-137  
checkNamespace(String, String), 1-94  
CLASSNOTFOUND, 3-6  
CLASSNOTFOUND\_MSG, 3-6  
cleanLobList(), 12-21  
cloneNode(boolean), 1-24, 1-57, 1-72, 1-94, 1-108  
close(), 12-7, 12-21  
collectTimingInfo(boolean), 12-21  
COMMA, 1-32  
Comment, 1-137  
comment(String), 1-11, 1-87  
CONN\_FILE, 3-6  
CONN\_FILE\_MSG, 3-6  
createAttribute(String), 1-72  
createAttribute(String, String), 1-37  
createBLOBTable(Connection, String), 4-67  
createCDATASection(String), 1-37, 1-73  
createComment(String), 1-38, 1-73  
createDocument(), 1-38  
createDocumentFragment(), 1-74  
createElement(String), 1-38, 1-74  
createEntityReference(String), 1-74  
createNestedRequest(URL, Dictionary), 3-24, 3-51  
createProcessingInstruction(String, String), 1-38, 1-75  
createTextNode(String), 1-39, 1-75  
createXMLTable(Connection, String), 4-68

## D

---

DATE\_FORMAT, 12-21  
DBAccess, 4-67  
DBAccess(), 4-67  
DBAccessBeanInfo, 4-74  
DBAccessBeanInfo(), 4-74  
DBViewer, 4-38  
DBViewer(), 4-39  
DBViewerBeanInfo, 4-53  
DBViewerBeanInfo(), 4-53  
debugPrintToFile(String, String), 3-18  
DEFAULT, 1-6  
DEFAULT\_BATCH\_SIZE, 12-21  
DefaultXMLDocumentHandler(), 1-10  
deleteBLOBName(Connection, String, String), 4-68  
deleteXML(Document), 12-21  
deleteXML(InputStream), 12-22  
deleteXML(Reader), 12-22  
deleteXML(String), 12-22  
deleteXML(URL), 12-23  
deleteXMLName(Connection, String, String), 4-68  
DictionaryOfParamsAsXMLDocument(Dictionary), 3-56  
doGet(HttpServletRequest, HttpServletResponse), 3-46  
DOMBuilder, 4-3  
DOMBuilder(), 4-4  
DOMBuilder(int), 4-4  
DOMBuilderBeanInfo, 4-14  
DOMBuilderBeanInfo(), 4-14  
domBuilderError(DOMBuilderEvent), 4-21  
domBuilderErrorCalled(DOMBuilderEvent), 4-18  
DOMBuilderErrorEvent, 4-16  
DOMBuilderErrorEvent(Object, Exception), 4-16  
DOMBuilderErrorListener, 4-18  
DOMBuilderEvent, 4-19  
DOMBuilderEvent(Object, int), 4-19  
DOMBuilderListener, 4-21  
domBuilderOver(DOMBuilderEvent), 4-21  
domBuilderStarted(DOMBuilderEvent), 4-21  
DOMParser(), 1-17  
doPost(HttpServletRequest, HttpServletResponse), 3-47

dropBLOBTable(Connection, String), 4-69  
dropXMLTable(Connection, String), 4-69  
DTD, 1-22  
    sql.query, 12-5  
DTDClassGenerator, 2-12  
DTDClassGenerator(), 2-12  
DTDName, 1-137

## E

---

ElemDeclName, 1-137  
ELEMENT, 1-32  
ELEMENTDECL, 1-107  
ElementDecl, 1-30  
elementdecl, 1-138  
ELEMENTS, 1-32  
EMPTY, 1-33  
EmptyElemTag, 1-138  
endDoctype(), 1-12, 1-87  
endElement(NSName), 1-12, 1-88  
ENTITIES, 1-6  
ENTITY, 1-6  
EntityDecl, 1-138  
EntityDeclName, 1-138  
EntityValue, 1-138  
ENUMERATION, 1-6  
ERR\_OUTPUT, 3-6  
ERR\_OUTPUT\_MSG, 3-7  
ERROR, 1-119  
ERROR\_TAG  
    sql.query, 12-5  
ERRORINCLUDING, 3-7  
ERRORINCLUDING\_MSG, 3-7  
ERRORLOADINGURL, 3-7  
ERRORLOADINGURL\_MSG, 3-7  
ERRORREADINGPARAM, 3-7  
ERRORREADINGPARAM\_MSG, 3-7  
ETag, 1-138  
ETagName, 1-138  
expectedElements(Element), 1-33, 1-76  
ExternalID, 1-138

## F

---

FATAL\_ERROR, 1-119

FATAL\_SHEETPOOL, 3-7  
FATAL\_SHEETPOOL\_MSG, 3-7  
finalize(), 12-24  
findAttrDecl(String), 1-33  
findElementDecl(String), 1-25  
findEntity(String, boolean), 1-25  
findNotation(String), 1-25  
FIXED, 1-7  
fontGet(AttributeSet), 4-55  
fontSet(MutableAttributeSet, Font), 4-55  
format(int, String), 3-12

## G

---

generate(DTD, String), 2-12  
generate(XMLSchema), 2-16  
getAttrDecls(), 1-34  
getAttribute(String), 1-95  
getAttributeNameFont(), 4-55  
getAttributeNameForeground(), 4-56  
getAttributeNode(String), 1-95  
getAttributes(), 1-96, 1-108, 2-31  
getAttributeValueFont(), 4-56  
getAttributeValueForeground(), 4-56  
getAttrPresence(), 1-8  
getAttrType(), 1-8  
getBackground(), 4-56  
getBLOBData(Connection, String, String), 4-69  
getCDATAFont(), 4-56  
getCDATAForeground(), 4-57  
getCGDocument(), 2-6  
getChildElements(), 2-10  
getChildNodes(), 1-25, 1-109  
getChildrenByTagName(String), 1-96  
getChildrenByTagName(String, String), 1-96  
getColumnNumber(int), 1-120  
getCommentDataFont(), 4-57  
getCommentDataForeground(), 4-57  
getConnectionName(), 3-24, 3-31  
getContentElements(), 1-34  
getContentType(), 1-34  
getCookie(String), 3-51  
getDefaultValue(), 1-8  
getDoctype(), 1-17, 1-76, 4-5  
getDocument(), 1-17, 4-5  
getDocumentElement(), 1-76  
getDTDNode(), 2-6  
getEditedText(), 4-57  
getElementDecls(), 1-26  
getElementsByTagName(String), 1-77, 1-97  
getElementsByTagName(String, String), 1-97  
getEncoding(), 1-77  
getEntities(), 1-26  
getEnumerationValues(), 1-8  
getErrorCode(), 12-34  
getErrorWriter(), 3-24, 3-32  
getException(), 4-16, 4-32  
getException(int), 1-120  
getExpandedName(), 1-40, 1-58, 1-97  
getExpandedName(int), 1-46  
getFirstChild(), 1-109  
getHostname(), 4-39  
getHttpServletRequest(), 3-51  
getHttpServletResponse(), 3-51  
getIcon(int), 4-14, 4-30, 4-53, 4-66, 4-74, 4-76, 4-81  
getID(), 4-20, 4-35  
getId(), 4-5, 4-26  
getImplementation(), 1-77  
getInstancename(), 4-39  
getInternalObj(), 12-7  
getJDBCCConnection(), 3-24, 3-32  
getJTextPane(), 4-57  
getLastChild(), 1-109  
getLength(), 1-46  
getLineNumber(int), 1-120  
getLocalName(), 1-40, 1-58, 1-98  
getLocalName(int), 1-46  
getMessage(), 4-17, 4-33  
getMessage(int), 1-120  
getMessageType(int), 1-121  
getMinimumSize(), 4-58  
getName(), 1-27, 1-59  
getName(int), 1-47  
getNameSize(), 4-70  
getNamespace(), 1-41, 1-59, 1-98  
getNamespace(int), 1-47  
getNextSibling(), 1-110  
getNodeAtOffset(int), 4-58  
getNodeName(), 1-110  
getNodeType(), 1-110





getXMLString(Node, int), 12-10  
getXMLStringFromSQL(String), 4-41  
getXMLTableNames(Connection, String), 4-71  
getXMLTreeModel(), 4-79  
getXslBuffer(), 4-42  
getXslCLOBFileName(), 4-42  
getXslCLOBTableName(), 4-42  
getXslFileName(), 4-42  
getXSQLConnection(), 3-26, 3-34

## H

---

handleAction(Node), 3-13  
hasChildNodes(), 1-28, 1-112  
HttpRequestAsXMLDocument(HttpServletRequest, String), 3-20

## I

---

ID, 1-7  
id, 4-35  
    DOMBuilderEvent, 4-19  
IDREF, 1-7  
IDREFS, 1-7  
ILLFORMEDXMLPARAMVAL, 3-7  
ILLFORMEDXMLPARAMVAL\_MSG, 3-7  
ILLFORMEDXMLRESOURCE, 3-7  
ILLFORMEDXMLRESOURCE\_MSG, 3-8  
IMPLIED, 1-7  
init(ServletConfig), 3-47  
init(XSQLPageRequest, Element), 3-14, 3-16  
inJServ(), 3-47  
inputDOMDocument, 4-54  
insertBefore(Node, Node), 1-112  
insertBLOBData(Connection, String, String, byte[]), 4-71  
insertXML(Document), 12-24  
insertXML(InputStream), 12-24  
insertXML(Reader), 12-24  
insertXML(String), 12-24  
insertXML(URL), 12-25  
insertXMLData(Connection, String, String, String), 4-71  
inSource, 4-3  
INSTANTIATIONERR, 3-8

INSTANTIATIONERR\_MSG, 3-8  
inStream, 4-3  
inString, 4-3  
INVALID\_URI, 3-8  
INVALID\_URI\_MSG, 3-8  
InvalidContentException, 2-14  
InvalidContentException(), 2-14  
InvalidContentException(String), 2-14  
INVALIDURL, 3-8  
INVALIDURL\_MSG, 3-8  
isEditable(), 4-60  
isIncludedRequest(), 3-26, 3-34  
isOracleDriver(), 3-26, 3-34  
isValidating, 2-4  
isXMLTable(Connection, String), 4-72

## J

---

jScrollPane, 4-54  
jTextPane, 4-55

## K

---

keepCursorState(boolean), 12-11  
keepObjectOpen(boolean), 12-11

## L

---

loadResBuffer(String), 4-42  
loadResBuffer(String, String), 4-43  
loadResBuffer(XMLDocument), 4-43  
loadResBufferFromClob(), 4-43  
loadResBufferFromFile(), 4-43  
loadXmlBuffer(String), 4-43  
loadXmlBuffer(String, String), 4-43  
loadXmlBuffer(XMLDocument), 4-44  
loadXmlBufferFromClob(), 4-44  
loadXmlBufferFromFile(), 4-44  
loadXMLBufferFromSQL(String), 4-44  
loadXslBuffer(String), 4-44  
loadXslBuffer(String, String), 4-44  
loadXslBuffer(XMLDocument), 4-45  
loadXslBufferFromClob(), 4-45  
loadXslBufferFromFile(), 4-45

## M

---

main(String[]), 1-44, 2-15, 3-17, 4-77  
MAXROWS\_ALL  
    sql.query, 12-6  
MAXROWS\_DEFAULT  
    sql.query, 12-6  
methodToCall, 4-3, 4-25  
MISSING\_ARGS, 3-8  
MISSING\_ARGS\_MSG, 3-8  
MISSING\_ATTR, 3-8  
MISSING\_ATTR\_MSG, 3-8  
MIXED, 1-33  
model, 4-78  
msg(String), 3-19

## N

---

NAMED\_CONN, 3-8  
NAMED\_CONN\_MSG, 3-9  
newDocument(), 3-39  
NMTOKEN, 1-7  
NMTOKENS, 1-7  
NO\_CONN, 3-9  
NO\_CONN\_DEF, 3-9  
NO\_CONN\_DEF\_MSG, 3-9  
NO\_CONN\_MSG, 3-9  
NO\_XSQL\_FILE, 3-9  
NO\_XSQL\_FILE\_MSG, 3-9  
NodeFactory, 1-36  
NodeFactory(), 1-37  
NOFUNCTIONNAME, 3-9  
NOFUNCTIONNAME\_MSG, 3-9  
NONE  
    sql.query, 12-6  
NOPOSTEDXML, 3-9  
NOPOSTEDXML\_MSG, 3-9  
NOQUERYSUPPLIED, 3-9  
NOQUERYSUPPLIED\_MSG, 3-10  
normalize(), 1-99  
NOTANACTIONHANDLER, 3-10  
NOTANACTIONHANDLER\_MSG, 3-10  
NOTATION, 1-7  
NotationDecl, 1-139  
NSName, 1-40

NSResolver, 1-42  
NULLPARAM, 3-10  
NULLPARAM\_MSG, 3-10

## O

---

OR, 1-33  
oracg, 2-15  
oracg(), 2-15  
Oracle9i Case Studies - XML Applications, xvi  
oracle.xml.async, 4-1  
oracle.xml.classgen, 2-1  
oracle.xml.parser.v2, 1-1  
OracleXMLQuery  
    (Connection, ResultSet), 12-6  
    (Connection, String), 12-7  
    (OracleXMLDataSet), 12-7  
    sql.query, 12-2  
OracleXMLSave, 12-18  
    (Connection, OracleColumnName[]), 12-21  
    (Connection, String), 12-21  
OracleXMLSQLException, 12-32  
    (Exception), 12-33  
    (Exception, String), 12-33  
    (String), 12-33  
    (String, Exception), 12-33  
    (String, Exception, String), 12-34  
    (String, int), 12-34  
    (String, int, String), 12-34  
    (String, String), 12-34  
OracleXMLSQLNoRowsException, 12-35  
OracleXMLSQLNoRowsException(), 12-36  
OracleXMLSQLNoRowsException(String), 12-36  
oracle.xml.xsql, 3-1  
oraxsl, 1-43  
oraxsl(), 1-44  
OWAXMLMALFORMED, 3-10  
OWAXMLMALFORMED\_MSG, 3-10

## P

---

parse(InputSource), 1-124, 4-6  
parse(InputStream), 1-124, 4-6  
parse(InputStream, URL, PrintWriter), 3-39  
parse(Reader), 1-124, 4-7

parse(Reader, PrintWriter), 3-39  
 parse(String), 1-125, 4-7  
 parse(URL), 1-125, 4-8  
 parse(URL, PrintWriter), 3-39  
 parseDocument(), 1-143  
 parseDTD(InputSource, String), 1-17, 4-8  
 parseDTD(InputStream, String), 1-18, 4-8  
 parseDTD(Reader, String), 1-18, 4-9  
 parseDTD(String, String), 1-19, 4-9  
 parseDTD(URL, String), 1-19, 4-10  
 parseFromString(String, PrintWriter), 3-39  
 parseFromString(StringBuffer, PrintWriter), 3-39  
 parseResBuffer(), 4-45  
 parseXmlBuffer(), 4-45  
 parseXslBuffer(), 4-45  
 PI, 1-139  
 PITarget, 1-139  
 PLUS, 1-33  
 POSTEDXML\_ERR, 3-10  
 POSTEDXML\_ERR\_MSG, 3-10  
 print(Document, PrintWriter), 3-39  
 print(OutputStream), 1-78, 1-113, 2-2  
 print(OutputStream, String), 1-79, 1-113, 2-3  
 print(PrintWriter), 1-79, 1-113  
 print(XMLOutputStream), 2-10  
 printAttributes(XMLOutputStream, String), 2-10  
 printDocumentMethods(), 2-12  
 printedErrorHeader(), 3-26, 3-34  
 printExternalDTD(OutputStream), 1-28, 1-80  
 printExternalDTD(OutputStream, String), 1-28, 1-80  
 printExternalDTD(PrintWriter), 1-29, 1-80  
 process(), 3-42  
 process(Dictionary), 3-43  
 process(Dictionary, PrintWriter, PrintWriter), 3-43  
 process(PrintWriter, PrintWriter), 3-43  
 processToDocument(Document, String, XSQLPageRequest), 3-54  
 processToWriter(Document, String, XSQLPageRequest), 3-54  
 processToXML(), 3-43  
 processToXML(Dictionary), 3-43  
 processToXML(Dictionary, PrintWriter), 3-44  
 processToXML(PrintWriter), 3-44  
 processXSL(XSLStylesheet, InputStream,

URL), 1-149, 4-27  
 processXSL(XSLStylesheet, Reader, URL), 1-150, 4-27  
 processXSL(XSLStylesheet, URL, URL), 1-150, 4-27  
 processXSL(XSLStylesheet, XMLDocument), 1-150, 4-28  
 processXSL(XSLStylesheet, XMLDocument, OutputStream), 1-152, 4-28  
 processXSL(XSLStylesheet, XMLDocument, PrintWriter), 1-152  
 processXSL(XSLStylesheet, XMLDocumentFragment), 1-151  
 processXSL(XSLStylesheet, XMLDocumentFragment, OutputStream), 1-151  
 processXSL(XSLStylesheet, java.io.OutputStream), 1-151  
 processXSL(XSLStylesheet, XMLDocumentFragment, PrintWriter), 1-152

## Q

---

QMARK, 1-33

## R

---

reader, 4-4  
 Reference, 1-139  
 releaseResource(), 4-23  
 removeAttribute(String), 1-100  
 removeAttributeNode(Attr), 1-100  
 removeChild(Node), 1-114  
 removeDOMBuilderErrorListener(DOMBuilderErrorListener), 4-10  
 removeDOMBuilderListener(DOMBuilderListener), 4-10  
 removeDOMTransformerErrorListener(XSLTransformerErrorListener), 4-28  
 removeXSLTParam(String), 12-11, 12-26  
 removeXSLTransformerListener(XSLTransformerListener), 4-29  
 replaceChild(Node, Node), 1-80, 1-114  
 replaceXMLData(Connection, String, String, String), 4-72  
 requestProcessed(), 3-26, 3-34  
 REQUIRED, 1-7  
 Res, 3-3

- Res(), 3-12
- resolveNamespacePrefix(String), 1-42, 1-100
- ResourceManager, 4-23
- ResourceManager(int), 4-23
- result, 4-4, 4-25
- rootName, 4-4
- ROW\_TAG
  - sql.query, 12-6
- ROWIDATTR\_TAG
  - sql.query, 12-6
- ROWSET\_TAG
  - sql.query, 12-6
- run(), 4-11, 4-29

## S

---

- safeURLAsString(URL), 3-56
- saveResBuffer(String), 4-46
- saveResBuffer(String, String), 4-46
- saveResBufferToClob(), 4-46
- saveResBufferToFile(), 4-46
- saveXmlBuffer(String), 4-46
- saveXmlBuffer(String, String), 4-46
- saveXmlBufferToClob(), 4-47
- saveXmlBufferToFile(), 4-47
- saveXslBuffer(String), 4-47
- saveXslBuffer(String, String), 4-47
- saveXslBufferToClob(), 4-47
- saveXslBufferToFile(), 4-47
- SAXAttrList, 1-45
- SAXParser(), 1-52
- SCHEMA
  - sql.query, 12-6
- SchemaClassGenerator, 2-16
- SchemaClassGenerator(), 2-16
- SchemaClassGenerator(String), 2-16
- scrollPane, 4-78
- select(Document, String), 3-56
- select(Element, String), 3-56
- select(XMLDocument, String), 3-56
- select(XMLElement, String), 3-57
- selectFirst(Document, String), 3-57
- selectFirst(Element, String), 3-57
- selectFirst(XMLDocument, String), 3-57
- selectFirst(XMLElement, String), 3-57
- selectNodeAt(int), 4-60
- selectNodes(String, NSResolver), 1-115
- selectSingleNode(String, NSResolver), 1-115
- setAttribute(String, String), 1-101, 2-6
- setAttributeNameFont(Font), 4-61
- setAttributeNameForeground(Color), 4-61
- setAttributeNode(Attr), 1-101
- setAttributeValueFont(Font), 4-61
- setAttributeValueForeground(Color), 4-61
- setBackground(Color), 4-61
- setBaseURL(URL), 1-126, 4-11
- setBatchSize(int), 12-26
- setCDATAFont(Font), 4-62
- setCDATAForeground(Color), 4-62
- setCollIdAttr(String), 12-11
- setCollIdAttrName(String), 12-11
- setCommentDataFont(Font), 4-62
- setCommentDataForeground(Color), 4-62
- setCommitBatch(int), 12-26
- setConnectionName(String), 3-26, 3-35
- setContentType(String), 3-26, 3-35, 3-52
- setDataHeader(Reader, String), 12-11
- setDateFormat(String), 12-12, 12-26
- setDebugMode(boolean), 4-11
- setDoctype(DTD), 1-12, 1-88, 1-126, 4-11
- setDocument(CGDocument), 2-6
- setDocumentHandler(DocumentHandler), 1-52
- setDTDHandler(DTDHandler), 1-53
- setEditable(boolean), 4-63
- setEncoding(String), 1-81, 12-12
- setEntityResolver(EntityResolver), 1-53
- setErrorHandler(ErrorHandler), 1-54, 1-143
- setErrorStream(OutputStream), 1-19, 1-143, 1-153, 4-11, 4-29
- setErrorStream(OutputStream, String), 1-20, 4-12
- setErrorStream(PrintWriter), 1-20, 4-12
- setErrorTag(String), 12-12, 12-34
- setException(Exception), 12-12
- setGenerateComments(boolean), 2-12, 2-17
- setHostname(String), 4-48
- setIgnoreCase(boolean), 12-27
- setIncludingRequest(XSQLPageRequest), 3-26, 3-35
- setInstancename(String), 4-48
- setJavaPackage(Vector), 2-13

setJavaPackage(XMLSchema, Vector), 2-17  
 setKeyColumnList(String[]), 12-27  
 setLocale(Locale), 1-81, 1-126, 1-153  
 setMaxRows(int), 12-13  
 setMetaHeader(Reader), 12-13  
 setNodeFactory(NodeFactory), 1-20, 4-12  
 setNodeValue(String), 1-61, 1-116, 2-11  
 setOutputDirectory(String), 2-13, 2-17  
 setPageEncoding(String), 3-26, 3-35, 3-52  
 setPageParam(String, String), 3-27, 3-35  
 setParam(String, String), 1-156  
 setPassword(String), 4-48  
 setPCDATAFont(Font), 4-63  
 setPCDATAForeground(Color), 4-63  
 setPIDataFont(Font), 4-63  
 setPIDataForeground(Color), 4-63  
 setPINameFont(Font), 4-64  
 setPINameForeground(Color), 4-64  
 setPort(String), 4-48  
 setPostedDocument(Document), 3-27, 3-36, 3-44  
 setPreserveWhitespace(boolean), 1-126, 4-13  
 setPrintedErrorHeader(boolean), 3-27, 3-36  
 setRaiseException(boolean), 12-13  
 setRaiseNoRowsException(boolean), 12-13  
 setResBuffer(String), 4-48  
 setResCLOBFileName(String), 4-49  
 setResCLOBTableName(String), 4-49  
 setResFileName(String), 4-49  
 setResHtmlView(boolean), 4-49  
 setResSourceEditView(boolean), 4-49  
 setResSourceView(boolean), 4-49  
 setResTreeView(boolean), 4-49  
 setRowIdAttrName(String), 12-14  
 setRowIdAttrValue(String), 12-14  
 setRowIdColumn(String), 12-14  
 setRowsetTag(String), 12-14  
 setRowTag(String), 12-14, 12-27  
 setSelectedNode(Node), 4-64  
 setSerializationMode(boolean), 2-13  
 setSkipRows(int), 12-14  
 setStandalone(String), 1-82  
 setStyleSheet(String), 12-15  
 setStyleSheetHeader(String), 12-15  
 setStyleSheetHeader(String, String), 12-15  
 setStyleSheetParameter(String, String), 3-27, 3-36  
 setStyleSheetURI(String), 3-27, 3-36  
 setSymbolFont(Font), 4-64  
 setSymbolForeground(Color), 4-64  
 setTagFont(Font), 4-65  
 setTagForeground(Color), 4-65  
 setTextDecl(String, String), 1-13, 1-88  
 setToken(int, boolean), 1-144  
 setTokenHandler(XMLToken), 1-144  
 setUpdateColumnList(String[]), 12-28  
 setUsername(String), 4-50  
 setValidationMode(boolean), 1-127, 2-13, 4-13  
 setValue(String), 1-62  
 setVersion(String), 1-82  
 setXmlBuffer(String), 4-50  
 setXmlCLOBFileName(String), 4-50  
 setXmlCLOBTableName(String), 4-50  
 setXMLDecl(String, String, String), 1-13, 1-89  
 setXMLDocument(Document), 4-65, 4-79  
 setXmlFileName(String), 4-50  
 setXmlSourceEditView(boolean), 4-51  
 setXmlSourceView(boolean), 4-51  
 setXmlTreeView(boolean), 4-51  
 setXslBuffer(String), 4-51  
 setXslCLOBFileName(String), 4-51  
 setXslCLOBTableName(String), 4-51  
 setXslFileName(String), 4-51  
 setXslSourceEditView(boolean), 4-52  
 setXslSourceView(boolean), 4-52  
 setXSLT(Reader, String), 12-15, 12-28  
 setXSLT(String, String), 12-15, 12-28  
 setXSLTParam(String, String), 12-16, 12-28  
 setXslTreeView(boolean), 4-52  
 showWarnings(boolean), 1-21, 1-153, 4-13, 4-29  
 SKIPROWS\_ALL  
     sql.query, 12-6  
 SKIPROWS\_DEFAULT  
     sql.query, 12-6  
 sleep(int), 4-24  
 splitText(int), 1-133  
 STag, 1-139  
 STagName, 1-139  
 startElement(NSName, SAXAttrList), 1-14, 1-89  
 storeID(String, String), 2-7  
 storeIDREF(String, String), 2-7  
 stringParamValue(Object), 3-57

## T

---

TextDecl, 1-139  
theTree, 4-79  
token(int, String), 1-140  
tokenize(InputSource), 1-144  
tokenize(InputStream), 1-144  
tokenize(Reader), 1-145  
tokenize(String), 1-145  
tokenize(URL), 1-146  
transformNode(XSLStylesheet), 1-116  
transformToDoc(), 4-52  
transformToRes(), 4-52  
transformToString(), 4-52  
translate(String, String), 3-57  
translate(URL, String), 3-57  
translateURL(String), 3-27, 3-36, 3-52  
type, 2-9

## U

---

UNHANDLED\_ERR, 3-10  
UNHANDLED\_ERR\_MSG, 3-10  
UNHANDLED\_ERR\_XSL\_PR, 3-10  
UNHANDLED\_ERR\_XSL\_PR\_MSG, 3-11  
UNHANDLED\_ERR\_XSL\_RD, 3-11  
UNHANDLED\_ERR\_XSL\_RD\_MSG, 3-11  
updateUI(), 4-79  
updateXML(Document), 12-29  
updateXML(InputStream), 12-29  
updateXML(Reader), 12-29  
updateXML(String), 12-30  
updateXML(URL), 12-30  
url, 4-4  
useConnectionPooling(), 3-27, 3-36  
useHTMLErrors(), 3-27, 3-37, 3-52  
useLowerCaseTagName(), 12-16  
useNullAttributeIndicator(boolean), 12-16  
useTypeForCollElemTag(boolean), 12-16  
useUpperCaseTagName(), 12-17

## V

---

validateContent(), 2-7  
validateContent(Element), 1-35  
validateElementContent(Element), 1-82

validEntity(String), 2-7  
validID(String), 2-8  
validNMTOKEN(String), 2-8  
valueOf(Element, String), 3-57  
valueOf(Node, String), 3-57  
valueOf(String, NSResolver), 1-116  
valueOf(XMLElement, String), 3-58  
valueOf(XMLNode, String), 3-58

## W

---

WARNING, 1-119

## X

---

XL(String, String), 3-21, 3-58  
XML\_INS\_ERR, 3-11  
XML\_INS\_ERR\_MSG, 3-11  
XML\_PARSE, 3-11  
XML\_PARSE\_MSG, 3-11  
XML\_SQL\_ERR, 3-11  
XML\_SQL\_ERR\_MSG, 3-11  
XMLAttr(String, String), 1-57  
XMLAttr(String, String, String, String), 1-57  
XMLCDATA, 1-63  
XMLCDATA(String), 1-65  
XMLComment, 1-66  
XMLComment(String), 1-68  
XMLDecl, 1-139  
XMLDocument, 1-69  
XMLDocument(), 1-72  
XMLDocumentFragment, 1-83  
XMLDocumentFragment(), 1-85  
XMLDocumentHandler, 1-86  
XMLElement, 1-91  
XMLElement(String), 1-94  
XMLElement(String, String, String), 1-94  
XMLEntityReference, 1-103  
XMLEntityReference(String), 1-104  
XMLNode, 1-105  
XMLParseException, 1-118  
XMLParseException(String, String, String, int, int,  
int), 1-120  
XMLParser, 1-122  
XMLPI, 1-128

- XMLPI(String, String), 1-130
- XMLSourceView, 4-54
- XMLSourceView(), 4-55
- XMLSourceViewBeanInfo, 4-66
- XMLSourceViewBeanInfo(), 4-66
- xmlStyledDocument, 4-55
- xmlTableExists(Connection, String), 4-73
- XMLText, 1-131
- XMLText(String), 1-133
- XMLToken, 1-135
- XMLTokenizer, 1-141
- XMLTokenizer(), 1-143
- XMLTokenizer(XMLToken), 1-143
- XMLTransformPanel, 4-75
- XMLTransformPanel(), 4-75
- XMLTransformPanelBeanInfo, 4-76
- XMLTransformPanelBeanInfo(), 4-76
- XMLTransViewer, 4-77
- XMLTransViewer(), 4-77
- XMLTreeView, 4-78
- XMLTreeView(), 4-79
- XMLTreeViewBeanInfo, 4-81
- XMLTreeViewBeanInfo(), 4-81
- XSL\_ERRORS, 3-11
- XSL\_ERRORS\_MSG, 3-11
- XSL\_NOFILE, 3-11
- XSL\_NOFILE\_MSG, 3-12
- XSL\_PARSE, 3-12
- XSL\_PARSE\_MSG, 3-12
- XSLException, 1-147
- XSLNOTFOUND, 3-12
- XSLNOTFOUND\_MSG, 3-12
- XSLProcessor(), 1-149
- XSLStylesheet, 1-154
- XSLStylesheet(InputStream, URL), 1-155
- XSLStylesheet(Reader, URL), 1-155
- XSLStylesheet(URL, URL), 1-155
- XSLStylesheet(XMLDocument, URL), 1-156
- XSLTransformer, 4-25
- XSLTransformer(), 4-25
- XSLTransformer(int), 4-25
- XSLTransformerBeanInfo, 4-30
- XSLTransformerBeanInfo(), 4-30
- xslTransformerError(XSLTransformerEvent), 4-37
- xslTransformerErrorCalled(XSLTransformerErrorEvent), 4-34
- XSLTransformerErrorEvent, 4-32
- XSLTransformerErrorEvent(Object, Exception), 4-32
- XSLTransformerErrorListener, 4-34
- XSLTransformerEvent, 4-35
- XSLTransformerEvent(Object, int), 4-35
- XSLTransformerListener, 4-37
- xslTransformerOver(XSLTransformerEvent), 4-37
- xslTransformerStarted(XSLTransformerEvent), 4-37
- XSQLActionHandler, 3-13
- XSQLActionHandlerImpl, 3-15
- XSQLActionHandlerImpl(), 3-16
- XSQLCommandLine, 3-17
- XSQLCommandLine(), 3-17
- XSQLDiagnostic, 3-18
- XSQLDiagnostic(String), 3-18
- XSQLHttpUtil, 3-20
- XSQLHttpUtil(), 3-20
- XSQLPageRequest, 3-22
- XSQLPageRequestImpl, 3-29
- XSQLPageRequestImpl(), 3-31
- XSQLPageRequestImpl(Hashtable), 3-31
- XSQLPageRequestImpl(String, Hashtable), 3-31
- XSQLParserHelper, 3-38
- XSQLParserHelper(), 3-39
- XSQLRequest, 3-40
- XSQLRequest(String), 3-42
- XSQLRequest(String, XSQLPageRequest), 3-42
- XSQLRequest(URL), 3-42
- XSQLRequest(URL, XSQLPageRequest), 3-42
- XSQLServlet, 3-46
- XSQLServlet(), 3-46
- XSQLServletPageRequest, 3-49
- XSQLServletPageRequest(HttpServletRequestRequest, HttpServletResponse), 3-50
- XSQLStylesheetProcessor, 3-53
- XSQLStylesheetProcessor(), 3-53
- XSQLUtil, 3-55
- XSQLUtil(), 3-56

