

uCosminexus Application Server

XML Processor User Guide

3021-3-J22-10(E)

Notices

■ Relevant program products

See the *Release Notes*.

■ Export restrictions

If you export this product, please check all restrictions (for example, Japan's Foreign Exchange and Foreign Trade Law, and USA export control laws and regulations), and carry out all required procedures.

If you require more information or clarification, please contact your Hitachi sales representative.

■ Trademarks

HITACHI, Cosminexus, TPBroker are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

AIX is a trademark of International Business Machines Corporation, registered in many jurisdictions worldwide.

Microsoft, Windows are trademarks of the Microsoft group of companies.

Microsoft, Windows Server are trademarks of the Microsoft group of companies.

Microsoft is a trademark of the Microsoft group of companies.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

UNIX is a trademark of The Open Group.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

■ Issued

Aug. 2022: 3021-3-J22-10(E)

■ Copyright

All Rights Reserved. Copyright (C) 2022, Hitachi, Ltd.

Preface

For details on the prerequisites before reading this manual, see the *Release Notes*.

■ Non-supported functionality

Some functionality described in this manual is not supported. Non-supported functionality includes:

- Audit log functionality
- Compatibility functionality
- Cosminexus Component Transaction Monitor
- Cosminexus Reliable Messaging
- Cosminexus TPBroker and VisiBroker
- Cosminexus Web Service - Security
- Cosminexus XML Security - Core functionality
- JP1 linkage functionality
- Management Server management portal
- Remote installation functionality for the UNIX edition
- SOAP applications complying with specifications other than JAX-WS 2.1
- uCosminexus OpenTP1 linkage functionality
- Virtualized system functionality
- XML Processor high-speed parse support functionality

■ Non-supported compatibility functionality

"Compatibility functionality" in the above list refers to the following functionality:

- Basic mode
- Check of JSP source compliance (cjjsp2java) with JSP1.1 and JSP1.2 specifications
- Database connection using Cosminexus DABroker Library
- EJB client application log subdirectory exclusive mode
- J2EE application test functionality
- Memory session failover functionality
- Servlet engine mode
- Simple Web server functionality
- Switching multiple existing execution environments
- Using EJB 2.1 and Servlet 2.4 annotation

Contents

Notices 2

Preface 3

1 An Overview of Cosminexus XML Processor 10

- 1.1 Product Features of Cosminexus XML Processor 11
 - 1.1.1 Enables Creation of Cross-platform Programs 11
 - 1.1.2 Enables Easy Parsing of XML Documents Using Java Programs 11
 - 1.1.3 Enables Easy Transformation of XML Documents Using Java Programs 12
 - 1.1.4 Improves the ease in development of XML-based applications 12
- 1.2 Positioning of Cosminexus XML Processor 14
- 1.3 Range of Features Supported by Cosminexus XML Processor 15
 - 1.3.1 Available functions such as the XML Parser and XSLT Transformer 15
 - 1.3.2 Character encodings that can be processed 17
 - 1.3.3 Shift_JIS Switch Function 17
 - 1.3.4 XSLTC Transformer Function 18
 - 1.3.5 Schema cache functionality 18
 - 1.3.6 High-speed parse support function 18
 - 1.3.7 Apache-specific functions 18

2 JAXP and JAXB Functionality 19

- 2.1 About JAXP 20
- 2.2 JAXP-defined Packages and Their Functionality 21
 - 2.2.1 javax.xml.parsers Package 21
 - 2.2.2 javax.xml.stream Package 22
 - 2.2.3 javax.xml.stream.events Package 23
 - 2.2.4 javax.xml.stream.util Package 23
 - 2.2.5 javax.xml.transform Package 23
 - 2.2.6 javax.xml.transform.dom Package 24
 - 2.2.7 javax.xml.transform.sax Package 24
 - 2.2.8 javax.xml.transform.stax Package 24
 - 2.2.9 javax.xml.transform.stream Package 24
 - 2.2.10 javax.xml.validation Package 24
 - 2.2.11 javax.xml.xpath Package 25
 - 2.2.12 javax.xml.namespace Package 26
 - 2.2.13 javax.xml.datatype Package 26
 - 2.2.14 javax.xml Package 27
 - 2.2.15 org.w3c.dom Package 27

2.2.16	org.w3c.dom.bootstrap Package	28
2.2.17	org.w3c.dom.ls Package	28
2.2.18	org.w3c.dom.events Package	29
2.2.19	org.xml.sax Package	30
2.2.20	org.xml.sax.ext Package	30
2.2.21	org.xml.sax.helpers Package	30
2.3	About JAXB	31
2.4	JAXB-defined Packages and Their Functionality	32
2.5	JAXB commands	33
2.5.1	csmxjc command (Binding from the XML Schema to Java)	33
2.5.2	csmschemagen command (Mapping from Java to the XML Schema)	36

3 Extended Functions of Cosminexus XML Processor 39

3.1	Shift_JIS Switch Function	40
3.2	XSLTC Transformer Function	42
3.2.1	An Overview of XSLTC Transformer	42
3.2.2	XSLTC Transformer-based System Development and Operation	42
3.2.3	Class Used by the XSLTC Transformer	43
3.2.4	How to use the XSLTC Transformer	44
3.3	Schema cache functionality	45
3.3.1	Overview of the schema cache functionality	45
3.3.2	Processing of the schema cache functionality	45
3.3.3	Setting up the schema cache	47
3.3.4	Using the schema cache	50
3.3.5	Deleting and resetting the schema cache	51
3.3.6	About using the most appropriate schema cache	55
3.4	High-speed parse support function	57
3.4.1	Overview of the high-speed parse support function	57
3.4.2	Flow of operations for high-speed parse	59
3.4.3	Creating the Pre-Parse XML document	60
3.4.4	Generating the preparsed object	68
3.4.5	Setting up the preparsed object	69
3.4.6	Parsing the XML document	69
3.4.7	Classes used in preparing the high-speed parse support function	70
3.4.8	Coding example for using the high-speed parse support function	75
3.4.9	Tuning the Pre-Parse XML document	76
3.4.10	Details of the tuning summary file	79
3.4.11	Details of the tuning details file	80
3.5	Apache-specific functions	84
3.5.1	Features that can be set up	84
3.5.2	Properties that can be set up	85
3.5.3	Namespace URIs that can be set up	85

3.6	Correspondence Between Error Messages in XML Schema and W3C Specifications	86
3.7	Parser Switching functionality	87
3.7.1	Overview of the Parser Switching functionality	87
3.7.2	Usage of the Parser Switching functionality	88
3.7.3	Performance of the Parser Switching functionality	90
3.7.4	Notes on the Parser Switching functionality	90
3.8	Secure processing functionality	93
3.8.1	Overview	93
3.8.2	Effective scope of the functionality	93
3.8.3	Items that can be restricted	94
3.8.4	Using the functionality	96
3.9	Compatibility option functionality	99
3.9.1	Overview of the compatibility option functionality	99
3.9.2	How to set up the compatibility option functionality	99

4 How to Use Features and Properties 101

4.1	How to Use the SAX2 Features and Properties	102
4.1.1	How to use the SAX2 features	102
4.1.2	How to use the SAX2 properties	103
4.2	How to use StAX properties	104
4.3	How to use XSLT features	105
4.4	How to Use the XML Schema Properties	106
4.4.1	How to Set the XML Schema Properties for the DOM Parser	106
4.4.2	How to Set the XML Schema Properties SAX Parser	106
4.4.3	How to Set the Schema Document in the XML Document	107
4.5	How to use the properties of the Shift_JIS switch function	109
4.5.1	How to Use the Shift_JIS Switch Function for the DOM Parser	109
4.5.2	How to Use the Shift_JIS Switch Function for the SAX Parser	110
4.5.3	How to Use the Shift_JIS Switch Function for the XSLT Transformer	111
4.6	How to use the properties of the high-speed parse support function	112
4.6.1	How to set up the preparsed object in DocumentBuilder	112
4.6.2	How to set up the preparsed object in SAXParser	113
4.6.3	How to set up the preparsed object in XMLReader	113
4.7	Using features and properties of the secure processing functionality	115
4.8	How to use the properties of JAXB	116

5 How to Create a Program 117

5.1	General Procedure for Creating a Program	118
5.1.1	Procedure for creating a program using Cosminexus XML Processor	118
5.2	Package Names Used by Cosminexus XML Processor	119
5.3	Troubleshooting Program Execution	120
5.4	Sample Program that Uses the DOM Parser	121

5.4.1	Processing Performed by the Sample Program	121
5.4.2	General Procedure for Creating the Sample Program	121
5.4.3	Sample program (SampleDOM.java)	122
5.4.4	Execution Result of the Sample Program (SampleDOM.java)	125
5.5	Sample Program that Uses the SAX Parser	126
5.5.1	Processing Performed by the Sample Program	126
5.5.2	General Procedure for Creating the Sample Program	126
5.5.3	XML Document to Use (SampleSAX.xml)	127
5.5.4	Sample Program (SampleSAX.java)	128
5.5.5	Execution Result of the Sample Program (SampleSAX.java)	130
5.6	Sample Program that Uses the XML Schema	131
5.6.1	Processing Performed by the Sample Program	131
5.6.2	XML Document to Use (purchaseOrder.xml and purchaseOrder-fail.xml)	132
5.6.3	Schema Documents to Use (purchaseOrder.xsd, personalData.xsd)	133
5.6.4	Sample Program When Using the DOM parser	137
5.6.5	Sample Program When Using the SAX Parser	139
5.7	Sample Program that Uses the XSLT Transformer	142
5.7.1	Processing Performed by the Sample Program	142
5.7.2	General Procedure for Creating the Sample Program	142
5.7.3	XML Document to Use (SampleXSLT.xml)	143
5.7.4	XSL File to Use (SampleXSLT.xsl)	144
5.7.5	Sample Program (SampleXSLT.java)	145
5.7.6	Execution Result of the Sample Program (SampleXSLT.java)	146
5.8	Sample Program that Uses the XSLTC Transformer	147

6 Notes on Using Cosminexus XML Processor 148

6.1	Notes common to JAXP1.4 functions	149
6.1.1	Range supported in the javax.xml.transform.stax.StAXSource class and the javax.xml.transform.stax.StAXResult class	150
6.1.2	Other notes	151
6.2	Notes on the DOM Parser	153
6.3	Notes on the SAX Parser	154
6.4	Notes on StAX	156
6.5	Notes on the Schema Validation	168
6.6	General Notes on XSLT and XSLTC	169
6.7	Notes on XSLT	172
6.7.1	Cases Where XSLT Does Not Report Errors	172
6.7.2	Other Notes	172
6.8	Notes on XSLTC	173
6.8.1	Notes on the Stylesheet Size	173
6.8.2	Notes on Transformation Performance	173
6.8.3	Notes on XSLT Elements	174

6.8.4	Notes on the XPath Expression	177
6.8.5	Cases where XSLTC Does Not Report Errors	177
6.8.6	Other Notes	179
6.9	Notes on the javax.xml.datatype Package	180
6.10	Notes on the javax.xml.validation Package	181
6.11	Notes on the javax.xml.xpath Package	183
6.12	Notes on the org.w3c.dom Package	185
6.13	Notes on the org.w3c.dom.bootstrap Package	187
6.14	Notes on the org.w3c.dom.ls Package	188
6.15	Notes on the org.xml.sax.ext Package	190
6.16	Notes on XInclude	191
6.17	Notes on implementation-dependent specifications	192
6.18	Precautions related to the schema cache functionality	194
6.18.1	Precautions related to performance	194
6.18.2	Precautions related to error output	194
6.18.3	Precautions related to schema definition files	194
6.18.4	Precautions related to reset up and deletion of a cache	195
6.19	Notes on the high-speed parse support function	196
6.19.1	Notes on setting up the preparsed object	196
6.19.2	Notes on the XML parser using the preparsed object	196
6.19.3	Notes on XML1.1	196
6.19.4	Notes on XInclude (high-speed parse support function)	196
6.19.5	Notes on how to specify an absolute path	197
6.19.6	Notes on the Pre-Parse XML document	197
6.19.7	Notes on tuning information	197
6.19.8	Notes on parse performance	198
6.20	Notes on JAXB	199
6.20.1	Common notes	199
6.20.2	Notes on schema compiler	199
6.20.3	Notes on schema generator	207
6.20.4	Notes on runtime	216
6.21	General XML Processor related notes	222

Appendixes 223

A	Differences Between Versions	224
A.1	Differences in Behaviors of XSLT Between Versions	224
A.2	Differences in the JAXB operation	224
A.3	Differences in the operation of the javax.xml.datatype package	226
A.4	Differences in the operation of the high-speed parse functionality	227
A.5	Differences in the operation of StAX	227
A.6	Differences in the operations of the Apache-specific functionality	228

A.7	Differences in Cosminexus 09-80 or later operations	228
B	Support Range of JAXB Specifications	229
B.1	Support range of the JAXB functions	229
B.2	Support range for JAXB characters	230
B.3	Support range of functions assumed as vendor-specific according to JAXB specifications	232
C	Efficiently using XML Processor	234
C.1	Basic functionality of XML Processor	234
C.2	Application functionality of XML Processor	236
C.3	Know-how for efficiently using XML Processor	237
D	Glossary	238

Index 239

1

An Overview of Cosminexus XML Processor

This chapter provides information you need to know before starting to use Cosminexus XML Processor, including the product's features, positioning, and supported features.

The range of supported features describes the available XML parser functions, XSLT transformer functions, JAXB data binding functions, and character codes that can be processed, and also introduces the extended functions of Cosminexus XML Processor.

1.1 Product Features of Cosminexus XML Processor

Cosminexus XML Processor is middleware that supports JAXP and JAXB, standard XML APIs for the Java language, and provides XML parser and XSLT transformer functions. Cosminexus XML Processor also provides the StAX streaming parser functionality and JAXB data binding functionality. The following subsections describe the product features of Cosminexus XML Processor.

1.1.1 Enables Creation of Cross-platform Programs

Cosminexus XML Processor supports JAXP and JAXB.

- JAXP (Java API for XML Processing) is a standard XML API for the Java language defined in JSR 206 of JCP (Java Community Process). JAXP includes APIs that generate XML parsers (DOM and SAX parsers) and XSLT transformers.
- JAXB (The Java Architecture for XML Binding) is a standard XML API defined in JSR 222 of Java Community Process. JAXB includes the schema compiler for converting from the XML Schema to Java, schema generator for converting from Java to the XML Schema, and APIs for exchanging data between XML documents and Java classes.

The standard XML APIs supported by Cosminexus XML Processor do not depend on implementation or coding of XML parsers, XSLT transformers, and JAXB data bindings. Therefore, Cosminexus XML Processor enables you to create cross-platform programs even if you are unaware of the implementation of the XML parser, XSLT transformer, and JAXB data binding used when you create a program that processes or performs operations on XML documents.

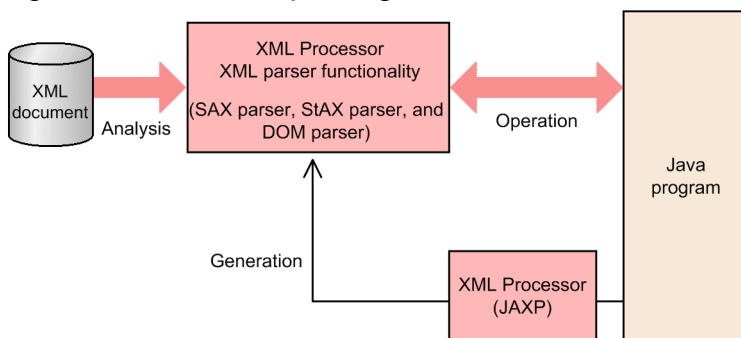
1.1.2 Enables Easy Parsing of XML Documents Using Java Programs

By providing a JAXP implementation, Cosminexus XML Processor enables Java programs to use XML parsers.

An XML parser is a library that parses XML documents and provides the parsed data to user programs in a manageable data structure. When you parse an XML document, tags and attributes in the document can also be validated for compliance with an XML Schema or DTD.

The following figure shows the process flow for parsing an XML document by using JAXP:

Figure 1–1: Flow of parsing XML documents in JAXP



By using JAXP, the Java program can generate an XML parser (SAX, StAX, or DOM parsers) for parsing XML documents.

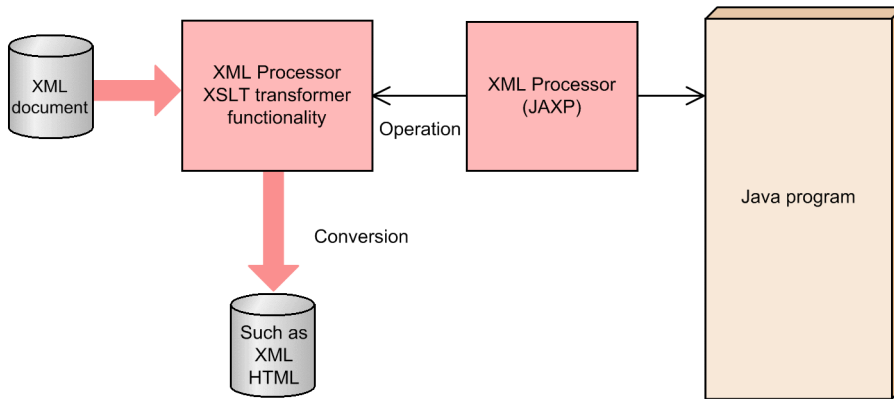
1.1.3 Enables Easy Transformation of XML Documents Using Java Programs

By providing a JAXP implementation, Cosminexus XML Processor enables Java programs to use XSLT transformers.

XSLT is the specification for transforming one XML document structure into another. The product of an XSLT implementation is called an *XSLT transformer*. An XSLT transformer reads, processes, and outputs XML documents. An XSLT transformer outputs not only XML documents but also HTML documents and texts.

The following figure shows the process flow for transforming an XML document by using JAXP. By using JAXP, a Java program can perform operations on the XML document transformation of the XSLT transformer.

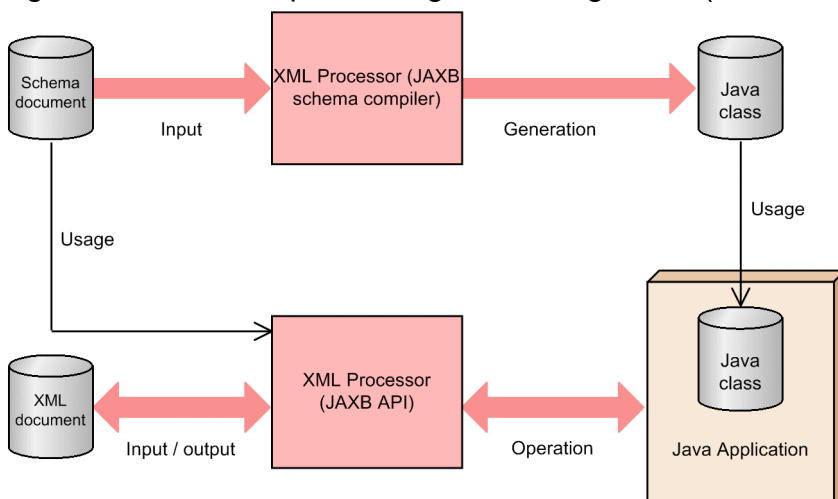
Figure 1–2: Flow of converting XML documents in JAXP



1.1.4 Improves the ease in development of XML-based applications

Cosminexus XML Processor provides the implementation of JAXB. When a schema document is input to the schema compiler, a Java class for accessing the XML document that is defined in the schema document is generated. If you use the created Java class, you can easily develop XML-based applications. The following figure shows the flow of processing XML using JAXB:

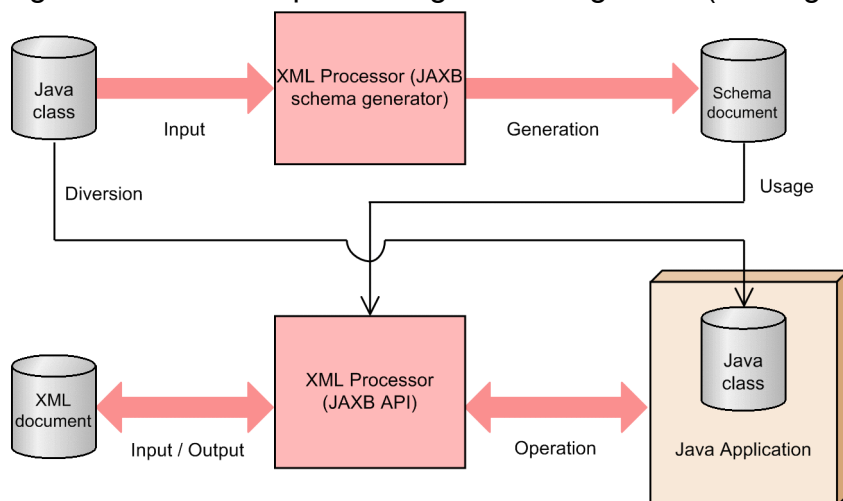
Figure 1–3: Flow of processing XML using JAXB (new development)



If the Java class is input to the schema generator, a schema document equivalent to the data structure of the Java class is generated. Therefore, you can easily build XML-based applications by leveraging the Java class that is an existing

resource. The following figure shows the flow of building an XML-based application in an existing Java class by using JAXB:

Figure 1–4: Flow of processing XML using JAXB (leveraging an existing Java class)



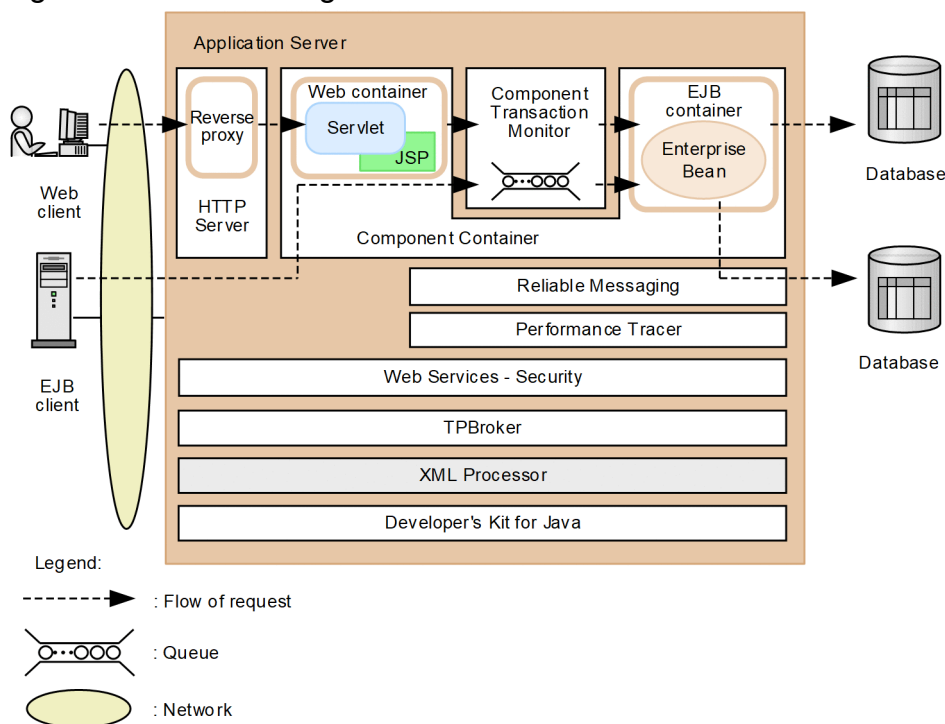
1.2 Positioning of Cosminexus XML Processor

This section describes the positioning of Cosminexus XML Processor in the Cosminexus Version 11 environment.

Cosminexus XML Processor is used in any phase that requires parsing, such as analyses, operation, creation, and XSLT transformation of XML documents by using Cosminexus Version 11. Combined with other Cosminexus Version 11 products, Cosminexus XML Processor can configure an environment that enables the integration of XML-based applications currently in demand, such as a SOAP communication infrastructure.

The following figure shows the positioning of Cosminexus XML Processor in the Cosminexus execution environment:

Figure 1–5: Positioning of Cosminexus XML Processor in the Cosminexus execution environment



1.3 Range of Features Supported by Cosminexus XML Processor

This section describes the features supported by Cosminexus XML Processor.

1.3.1 Available functions such as the XML Parser and XSLT Transformer

Cosminexus XML Processor supports JAXP1.4 and JAXB2.2 specifications. You can use the XML parser and XSLT transformer functions available through JAXP within the specifications defined in JAXP as shown below.

Important note

The JAXP versions have slightly different specifications. Similarly, note that JAXB 2.2 has slightly different specifications than its predecessor, JAXB 2.1.

For details about the support range of JAXB, see [Appendix B Support Range of JAXB Specifications](#).

(1) Specifications defined in JAXP

- JSR 206 Java API for XML Processing (JAXP) 1.4
For details on how to use the APIs, see the following page:
<http://docs.oracle.com/javase/6/docs/api/>
- Extensible Markup Language (XML)
 - Extensible Markup Language (XML) 1.1
<http://www.w3.org/TR/xml11>
 - XML 1.1 First Edition Specification Errata
<http://www.w3.org/XML/xml-V11-1e-errata>
 - Extensible Markup Language (XML) 1.0 (Third Edition)
<http://www.w3.org/TR/2004/REC-xml-20040204/>
 - Extensible Markup Language (XML) 1.0 (Third Edition) Errata
<http://www.w3.org/XML/xml-V10-3e-errata>
- Namespaces in XML
 - Namespaces in XML 1.1
<http://www.w3.org/TR/xml-names11/>
 - Namespaces in XML 1.1 Errata
<http://www.w3.org/XML/2004/xml-names11-errata>
 - Namespaces in XML 1.0
<http://www.w3.org/TR/REC-xml-names>
 - Namespaces in XML 1.0 Errata
<http://www.w3.org/XML/xml-names-19990114-errata>
- XML Schema
 - XML Schema Part 1: Structures Second Edition
<http://www.w3.org/TR/xmlschema-1/>

- XML Schema Part 1: Structures Second Edition Errata
<http://www.w3.org/2004/03/xmlschema-errata#Errata1>
- XML Schema Part 2: Datatypes Second Edition
<http://www.w3.org/TR/xmlschema-2/>
- XML Schema Part 2: Datatypes Second Edition Errata
<http://www.w3.org/2004/03/xmlschema-errata#Errata2>
- XSL Transformations (XSLT)
 - XSL Transformations (XSLT) Version 1.0
<http://www.w3.org/TR/xslt>
 - XSL Transformations (XSLT) Version 1.0 Errata
<http://www.w3.org/1999/11/REC-xslt-19991116-errata>
- XML Path Language (XPath)
 - XML Path Language (XPath) Version 1.0
<http://www.w3.org/TR/xpath>
 - XML Path Language (XPath) Version 1.0 Errata
<http://www.w3.org/1999/11/REC-xpath-19991116-errata>
- XML Inclusions (XInclude)
 - XML Inclusions (XInclude) Version 1.0
<http://www.w3.org/TR/xinclude/>
- Document Object Model (DOM) Level 3
 - Document Object Model (DOM) Level 3 Core
<http://www.w3.org/TR/DOM-Level-3-Core>
 - Document Object Model (DOM) Level 3 Load and Save
<http://www.w3.org/TR/DOM-Level-3-LS>
- Simple API for XML (SAX)
 - Simple API for XML (SAX) 2.0.2 (sax2r3)
<http://sax.sourceforge.net/>
 - Simple API for XML (SAX) 2.0.2 (sax2r3) Extensions
<http://sax.sourceforge.net/?selected=ext>
- Streaming API for XML (StAX)
 - Streaming API for XML (StAX) Version 1.0
<http://jcp.org/en/jsr/detail?id=173>

(2) Specifications defined in JAXB

- JSR 222 The JavaArchitecture for XML Binding 2.2
For details on how to use the APIs, see the following page:
<http://jcp.org/aboutJava/communityprocess/mrel/jsr222/index2.html>

1.3.2 Character encodings that can be processed

The character encoding for an XML document is specified in the encoding attribute (*XXX* in `encoding="XXX"`) of the XML declaration in an XML document. The character encoding can also be specified as the encoding of the `InputSource` object that is used as the input source for the XML document.

Cosminexus XML Processor can process the following character encodings among those registered by IANA. For the definition of a character encoding, see the document regarding the IANA character sets.

- UTF-8
- UTF-16
- UTF-16BE
- UTF-16LE
- Shift_JIS^{#1}
- Windows-31J
- ISO-2022-JP
- EUC-JP
- US-ASCII
- ISO-10646-UCS-2
- ISO-10646-UCS-4^{#2}
- ISO-8859-1

#1

For details about switching character encoding (SJIS or MS932) applied when using Shift_JIS, see [1.3.3 Shift_JIS Switch Function](#).

#2

Only the formats with the byte order of big-endian or little-endian and with no Byte-Order-Marks added are supported.



Important note

If any other character encoding is used, operation is not guaranteed.

1.3.3 Shift_JIS Switch Function

The Shift_JIS switch function changes the character encoding applied by Cosminexus XML Processor when the encoding attribute in the XML document is specified as Shift_JIS. Character encoding can be switched for each instance of the XML parser or XSLT transformer.

Character encoding specified for encoding of the XML document can be switched to SJIS (x-sjis-jdk1.1.7) or MS932 (x-sjis-cp932) according to the specification in the Shift_JIS switch function provided by Cosminexus XML Processor. For details about the correspondence between the XML encoding specification and the applied character encoding, see [Table 3-1](#) in [3.1 Shift_JIS Switch Function](#).

For details about how to use the Shift_JIS switch function, see [3.1 Shift_JIS Switch Function](#).

1.3.4 XSLTC Transformer Function

The XSLTC transformer offers the same functionality as the XSLT transformer provided by Cosminexus XML Processor, with improved performance for XML document transformation. Cosminexus XML Processor enables you to develop applications that use both XSLT and XSLTC transformers.

For details about how to use the XSLTC transformer function, see [3.2.4 How to use the XSLTC Transformer](#).

1.3.5 Schema cache functionality

The *schema cache functionality* caches results (grammar objects) of parsing schema documents that are defined with an XML Schema, in the memory or on the disk beforehand. You can use the schema cache functionality to substitute a processing that converts schema documents to grammar objects with referencing the cache, for improving performance of validation.

You must code the schema documents to be cached using the setup file (schema definition file).

For details about how to use the schema cache functionality, see the section [3.3 Schema cache functionality](#).

1.3.6 High-speed parse support function

The *high-speed parse support function* improves the execution speed of the parsing process by studying the features of the XML document to be parsed through pre-parsing, and thereafter using the study results when parsing the XML document.

For details about how to use the high-speed parse support function, see [3.4 High-speed parse support function](#).

1.3.7 Apache-specific functions

In version 08-70 and later versions of Cosminexus XML Processor, you can use Apache-specific features, properties, and Namespace URI in addition to the JAXP standards. These specific functions can be used with XML parsers and XSLT or XSLTC transformers. However, operations are not guaranteed because these functions are not covered by the JAXP standards. Therefore, you need to take measures such as adequately checking operations beforehand.

For details on the Apache-specific functions, see [3.5 Apache-specific functions](#).

2

JAXP and JAXB Functionality

This chapter gives an overview of JAXP and JAXB. You will need to understand this overview if you wish to use JAXP and JAXB when creating programs.

2.1 About JAXP

JAXP (Java API for XML Processing) is a standard XML API for the Java language defined in JSR 206 of the Java Community Process.

The following table describes the JAXP functions that Cosminexus XML Processor supports.

Table 2–1: JAXP functions that Cosminexus XML Processor supports

Function	Overview
DOM	Analyze an XML document to generate a DOM tree. Also, manipulates the generated DOM tree.
SAX	Analyze an XML document to generate an SAX event. Also, handles the generated event.
StAX	Parses an XML document to generate a StAX event and also processes the generated event. Does not require a handler, and therefore can perform procedural processing of XML documents.
XSLT	Receives an XML document as input, transforms the document based on a stylesheet, and then outputs another XML document, an HTML document, or text.
XPath	Evaluates an XPath expression.
Validation	Validates an XML document based on the schema document.
Datatype	Processes the date and time format data defined in W3C XML Schema 1.0.

JAXP includes APIs that can generate DOM parsers, SAX parsers, StAX parsers, XSLT transformers, XPath objects, Validation objects, and Datatype objects. These APIs do not depend on the JAXP implementation. This frees you from needing to be aware of the implementation of the XML processor used when you create a program that handles and operates XML documents. Therefore, Cosminexus XML Processor enables you to create cross-platform programs.

2.2 JAXP-defined Packages and Their Functionality

JAXP defines packages for using the functions listed in [Table 2-1](#) of *2.1 About JAXP*. The following table gives an overview of APIs that are included in the packages defined in JAXP:

Table 2–2: An overview of APIs included in the packages defined in JAXP

Package	An overview of APIs included in the packages
<code>javax.xml.parsers</code>	APIs for performing parsing with a DOM parser and SAX parser.
<code>javax.xml.stream</code>	APIs for StAX parsing.
<code>javax.xml.stream.events</code>	
<code>javax.xml.stream.util</code>	
<code>javax.xml.transform</code>	APIs for XSLT transformers
<code>javax.xml.transform.dom</code>	APIs that use DOM for inputting to and outputting from XSLT transformers
<code>javax.xml.transform.sax</code>	APIs that use SAX for inputting to and outputting from XSLT transformers
<code>javax.xml.transform.stax</code>	APIs that use StAX for performing input and output from the XSLT transformers
<code>javax.xml.transform.stream</code>	APIs that use streams for inputting to and outputting from XSLT transformers.
<code>javax.xml.validation</code>	APIs for validating XML documents
<code>javax.xml.xpath</code>	APIs for evaluating XPath expressions
<code>javax.xml.namespace</code>	API for processing XML namespaces
<code>javax.xml.datatype</code>	APIs for processing the date and time format data defined in W3C XML Schema 1.0
<code>javax.xml</code>	Character-string constants related to XML
<code>org.w3c.dom</code>	APIs for manipulating DOM trees
<code>org.w3c.dom.bootstrap</code>	APIs for obtaining a DOM implementation
<code>org.w3c.dom.ls</code>	APIs for loading and saving XML documents
<code>org.w3c.dom.events</code>	APIs for handling DOM events
<code>org.xml.sax</code>	API for basic SAX processing
<code>org.xml.sax.ext</code>	APIs for extended SAX processing
<code>org.xml.sax.helpers</code>	Sets of helper classes for SAX processing

The following subsections give an overview of each package.

2.2.1 javax.xml.parsers Package

The `javax.xml.parsers` package provides APIs for parsing with a DOM parser and SAX parser.

Figures 2-1 and 2-2 show the flow of processing for applications that perform a DOM parse analysis and a SAX parse analysis respectively.

Figure 2–1: Flow of processing for applications that perform a DOM parse

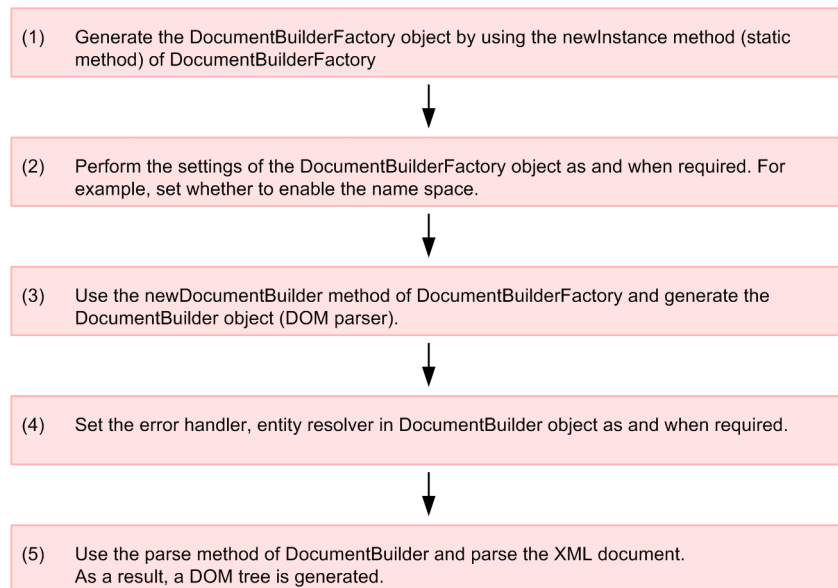
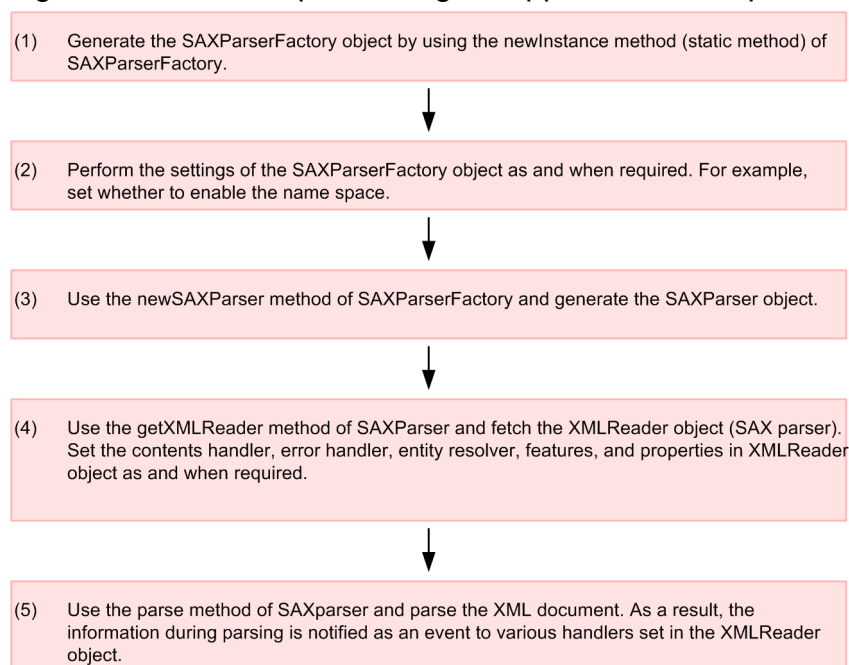


Figure 2–2: Flow of processing for applications that perform a SAX parse



It is also possible to generate a SAX parser with the XMLReaderFactory class.

For details on the `javax.xml.parsers` package, see *Chapter 7. Package javax.xml.parsers* in the documentation for JSR 206 Java API for XML Processing (JAXP) 1.4.

For details on the coding samples that perform a DOM parse, see *5.4 Sample Program that Uses the DOM Parser*. For details on the coding samples that perform a SAX parse, see *5.5 Sample Program that Uses the SAX Parser*.

2.2.2 javax.xml.stream Package

The `javax.xml.stream` package provides APIs for StAX parsing.

For details on the `javax.xml.stream` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.3 javax.xml.stream.events Package

The `javax.xml.stream.events` package provides APIs for processing the StAX events.

For details on the `javax.xml.stream.events` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.4 javax.xml.stream.util Package

The `javax.xml.stream.util` package provides utility APIs for StAX processing.

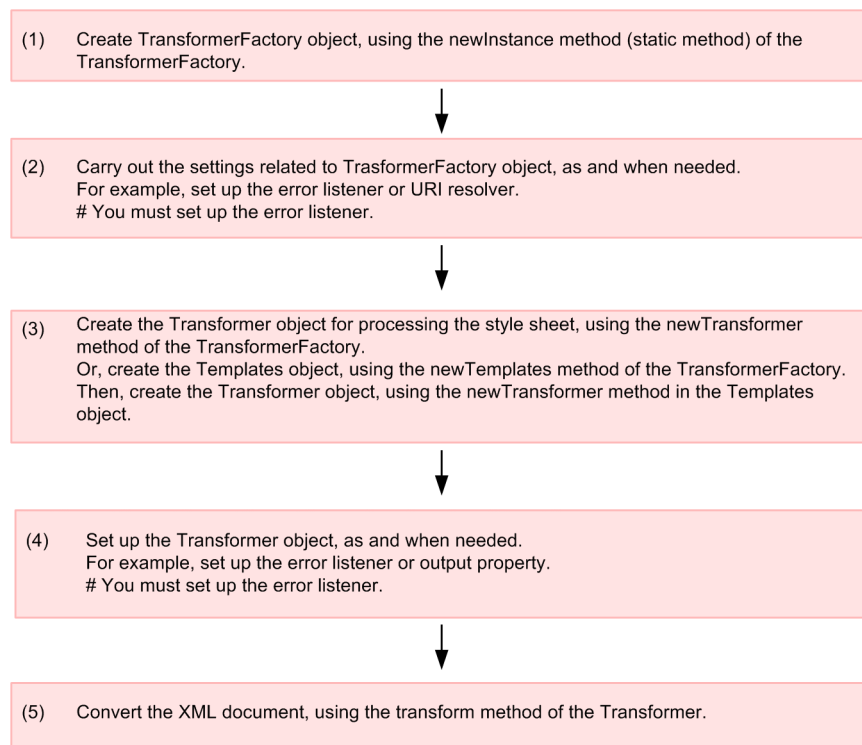
For details on the `javax.xml.stream.util` package, see the *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.5 javax.xml.transform Package

The `javax.xml.transform` package provides APIs for XSLT transformers.

The following figure shows the flow of processing for applications that perform XSLT-based transformation.

Figure 2–3: Flow of processing for applications that perform XSLT-based transformation



For details on the `javax.xml.transform` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

For details on the coding samples for XSLT-based transformation, see [5.7 Sample Program that Uses the XSLT Transformer](#).

2.2.6 javax.xml.transform.dom Package

The `javax.xml.transform.dom` package provides APIs that use DOM for inputting to and outputting from XSLT transformers.

For details on the `javax.xml.transform.dom` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.7 javax.xml.transform.sax Package

The `javax.xml.transform.sax` package provides APIs that use SAX for inputting to and outputting from XSLT transformers.

For details on the `javax.xml.transform.sax` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.8 javax.xml.transform.stax Package

The `javax.xml.transform.stax` package provides APIs for using StAX for input and output of XSLT transformers.

For details on the `javax.xml.transform.stax` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.9 javax.xml.transform.stream Package

The `javax.xml.transform.stream` package provides APIs that use stream for inputting to and outputting from XSLT transformers.

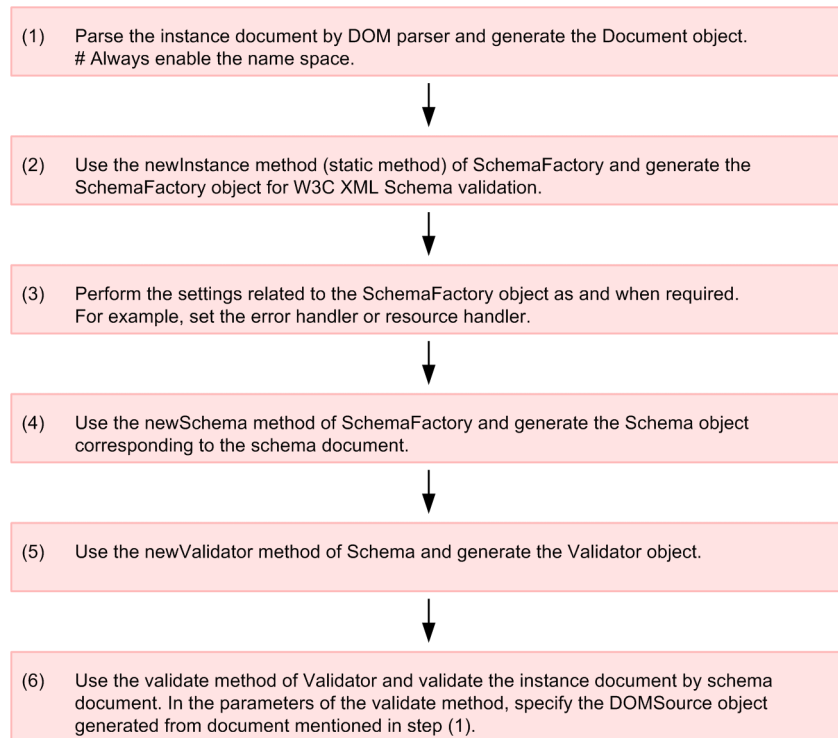
For details on the `javax.xml.transform.stream` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.10 javax.xml.validation Package

The `javax.xml.validation` package provides APIs for validating XML documents.

The following figure shows the flow of processing for applications that validate XML documents by using a class included in the `javax.xml.validation` package.

Figure 2–4: Flow of processing for applications that validate XML documents by using a class included in the `javax.xml.validation` package



In addition, with a DOM parser, analysis and validation can be performed simultaneously. With a SAX parser, analysis and validation can also be performed simultaneously.

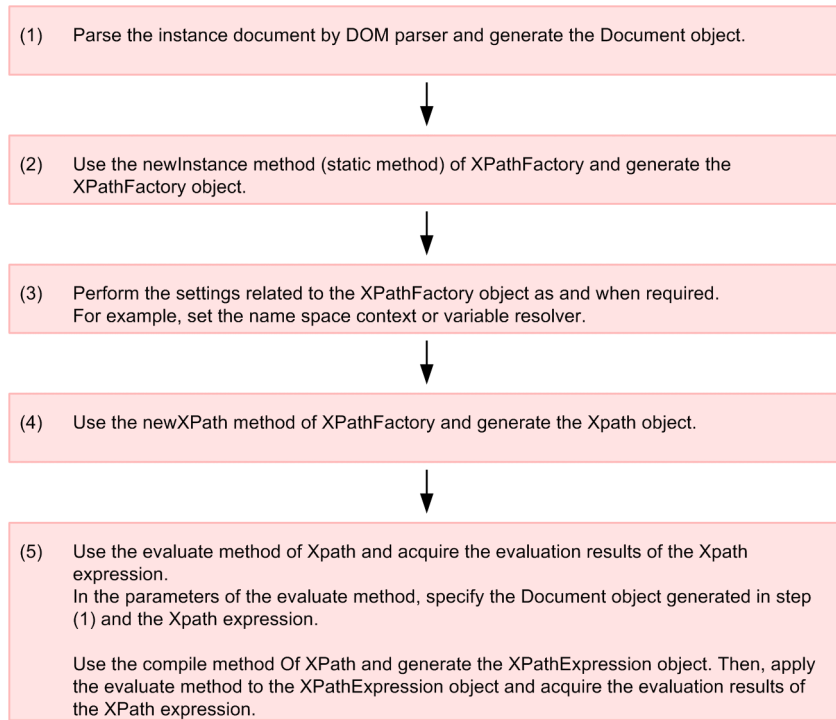
For details on the `javax.xml.validation` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.11 javax.xml.xpath Package

The `javax.xml.xpath` package provides APIs for evaluating XPath expressions.

The following figure shows the flow of processing for applications that evaluate XPath expressions.

Figure 2–5: Flow of processing for applications that evaluate XPath expressions



It is also possible to evaluate XPath expressions by using an `InputSource` object as an input.

For details on the `javax.xml.xpath` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.12 javax.xml.namespace Package

The `javax.xml.namespace` package provides APIs for processing XML namespaces. This package is only used as arguments or return values for APIs included in the `javax.xml.xpath` and `javax.xml.datatype` packages.

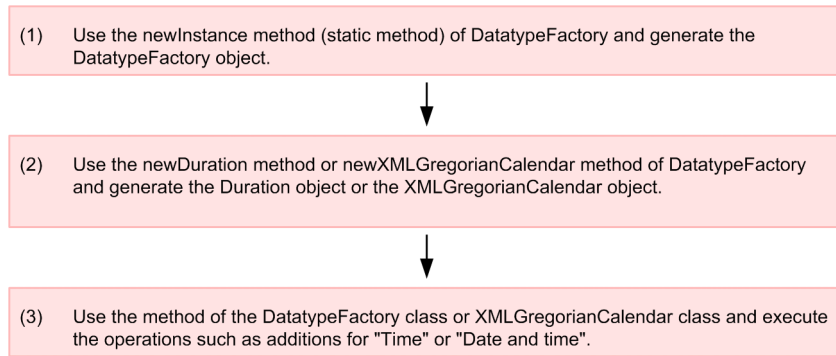
For details on the `javax.xml.namespace` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.13 javax.xml.datatype Package

The `javax.xml.datatype` package provides APIs for processing date and time format data that is defined in W3C XML Schema 1.0.

The following figure shows the flow of processing for applications that process date and time format data.

Figure 2–6: Flow of processing for applications that process date and time format data



For details on the `javax.xml.datatype` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.14 javax.xml Package

The `javax.xml` package defines the character-string constants related to XML.

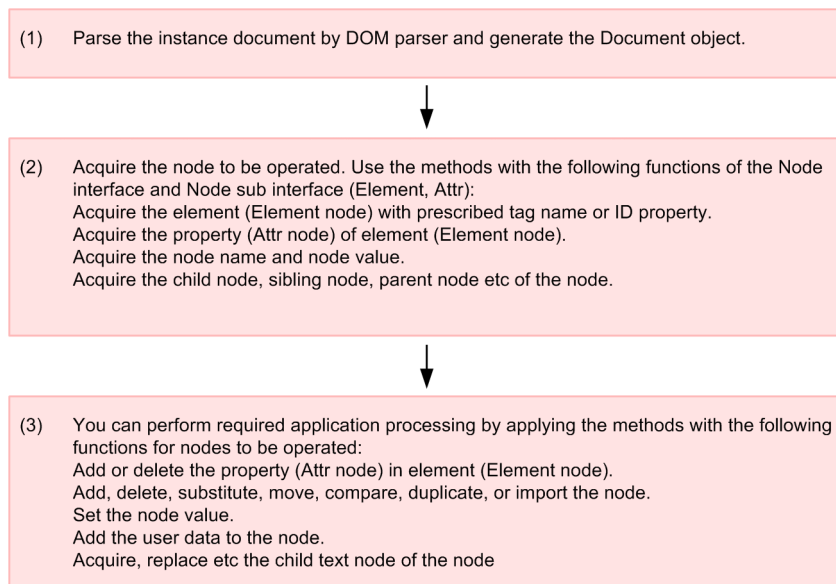
For details on the `javax.xml` package, see *Javadoc* in the *JSR 206 Java API for XML Processing (JAXP) 1.4*.

2.2.15 org.w3c.dom Package

The `org.w3c.dom` package provides APIs for manipulating DOM trees.

The following figure shows the flow of processing for applications that manipulate DOM trees.

Figure 2–7: Flow of processing for applications that manipulate DOM trees



For details on the `org.w3c.dom` package, see the following sections in the documentation *W3C Document Object Model (DOM) Level 3 Core Specification*:

- *1.4 Fundamental Interfaces: Core Module*

- *1.5 Extended Interfaces: XML Module*
- *G: Java Language Binding G.2 Other Core interfaces*

For details on a coding sample for manipulating DOM trees, see [5.4 Sample Program that Uses the DOM Parser](#).

2.2.16 org.w3c.dom.bootstrap Package

The `org.w3c.dom.bootstrap` package provides APIs for obtaining a DOM implementation.

For details on the `org.w3c.dom.bootstrap` package, see the following sections in the documentation *W3C Document Object Model (DOM) Level 3 Core Specification*:

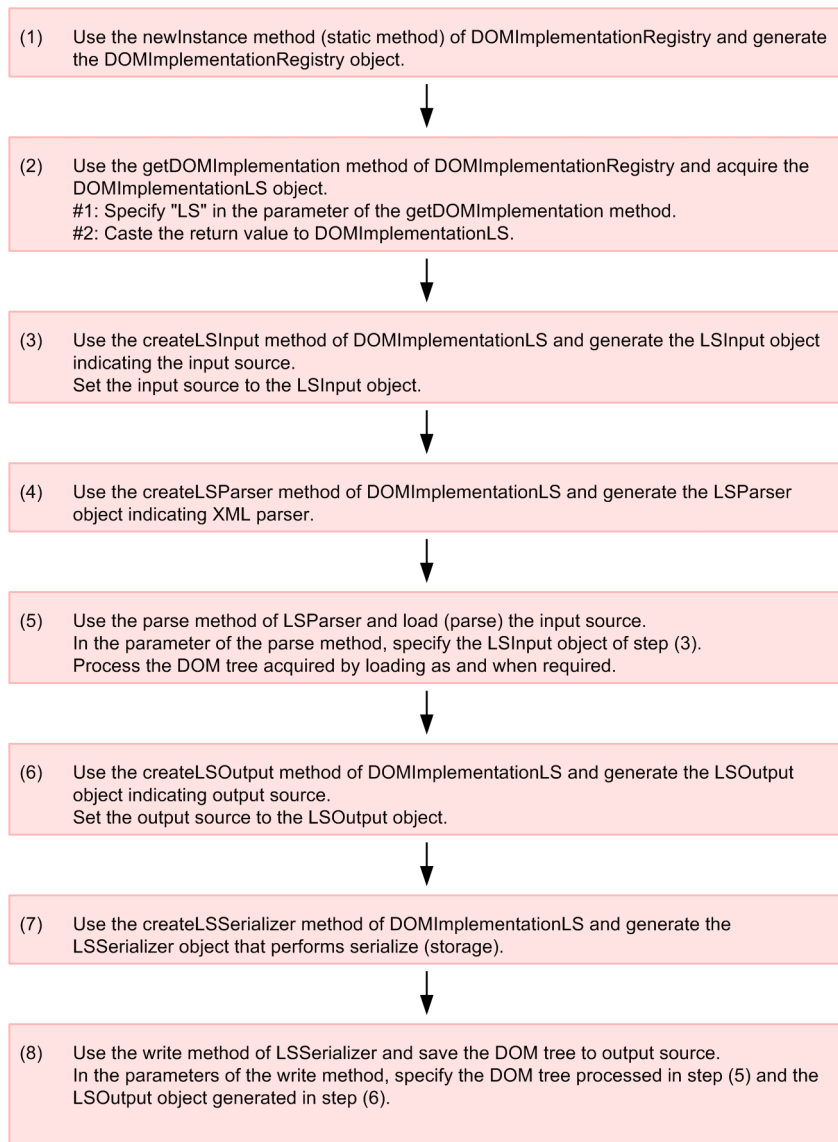
- *1.3.7 Bootstrapping*
- *G: Java Language Binding G.1 Java Binding Extension*

2.2.17 org.w3c.dom.ls Package

The `org.w3c.dom.ls` package provides APIs for loading and saving XML documents.

The following figure shows the flow of processing for applications that load and save XML documents.

Figure 2–8: Flow of processing for applications that load and save XML documents



For details on the `org.w3c.dom.ls` package, see the following sections in the documentation *W3C Document Object Model (DOM) Level 3 Load and Save Specification*:

- *1.3 Fundamental Interfaces*
- *B: Java Language Binding*

2.2.18 org.w3c.dom.events Package

The `org.w3c.dom.events` package is used by asynchronous `LSParser` objects that implement the `org.w3c.dom.ls.LSParser` interface. This package is not used because Cosminexus XML Processor does not support asynchronous `LSParser` objects.

For details on the `org.w3c.dom.events` package, see the following sections in the documentation *W3C Document Object Model (DOM) Level 3 Events Specification*:

- *1.6 Basic interfaces*

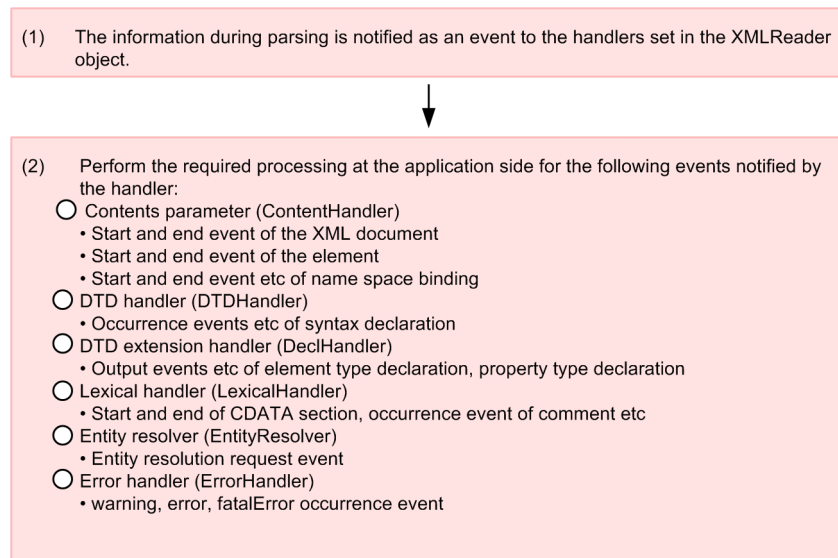
- 1.7 Event module definitions
- D: Java Language Binding

2.2.19 org.xml.sax Package

The `org.xml.sax` package provides APIs for basic SAX processing.

The following figure shows the flow of processing for applications that handle SAX events.

Figure 2–9: Flow of processing for applications that handle SAX events



For details on the `org.xml.sax` package, see the *Javadoc of Simple API for XML (SAX) 2.0.2 (sax2r3)*.

For a coding sample for handling SAX events, see [5.5 Sample Program that Uses the SAX Parser](#).

2.2.20 org.xml.sax.ext Package

The `org.xml.sax.ext` package provides APIs for extended SAX processing.

For details on the `org.xml.sax.ext` package, see the *Javadoc of Simple API for XML (SAX) 2.0.2 (sax2r3) Extensions*.

2.2.21 org.xml.sax.helpers Package

The `org.xml.sax.helpers` package provides a set of helper classes for SAX processing.

For details on the `org.xml.sax.helpers` package, see the *Javadoc documentation for Simple API for XML (SAX) 2.0.2 (sax2r3) Extensions*.

2.3 About JAXB

JAXB (The Java Architecture for XML Binding) is a standard XML API for the Java language defined in JSR 222 of Java Community Process.

For details on the JAXB functions provided by Cosminexus XML Processor, see [*Appendix B Support Range of JAXB Specifications*](#).

JAXB contains APIs for generating Marshaller and Unmarshaller. These APIs do not depend on the implementation of JAXB. JAXB also contains a schema compiler and schema generator. The Java class and schema document generated by the schema compiler and schema generator are not dependent on the implementation of JAXB. Thus, you need not be aware of the implementation of the XML processor used, when creating a program for processing or operating an XML document. Therefore, cross-platform programs can be created.

2.4 JAXB-defined Packages and Their Functionality

JAXB provides the packages for using the functions. The following table gives an overview of APIs that are included in the packages defined in JAXB.

Table 2–3: Overview of APIs included in the packages defined in JAXB

Package	An overview of APIs included in the packages
<code>javax.xml.bind</code>	APIs for runtime binding frameworks, such as Marshal and Unmarshal.
<code>javax.xml.bind.annotation</code>	APIs for customizing the mapping from a Java class to a schema document.
<code>javax.xml.bind.annotation.adapters</code>	APIs for the adapter class used when any Java class is used in JAXB.
<code>javax.xml.bind.attachment</code>	APIs for performing Marshal and Unmarshal of the MIME-format binary data.
<code>javax.xml.bind.util</code>	APIs for the utilities of JAXB.

For details on the packages, see the *Javadoc* of *JSR 222 The Java Architecture for XML Binding 2.2*.

2.5 JAXB commands

Concrete usage methods of the schema compiler and schema generator are not provided in JSR 222. With Cosminexus XML Processor, you can use the schema compiler and schema generator by using the commands described in this section.

- Symbols used for describing the command syntax

In this manual, the following symbols are used for coding the format of commands:

Symbol	Convention
[]	Items enclosed within this symbol may be omitted.
<>	Items enclosed within this symbol specify variable values such as a service location and file.

- Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path

The rules for specifying the file name or the directory name of a command are as follows:

- The characters that can be specified in the file name or directory name of a command are alphanumeric characters and the following symbols:
@ & = + \$, - _ . ! ~ ' ()
- The characters that you can use as path delimiting characters are as follows:
 \ (In Windows)
 / (In UNIX)
- Path can be specified either as an absolute path or as a relative path.
- Commands provided with XML Processor

The following table describes the commands provided with Cosminexus XML Processor as the command line interfaces of the schema compiler and schema generator.

Table 2–4: List of commands

File name of command		Function
Windows	UNIX	
csmxjc.bat	csmxjc	Schema compiler for binding from the XML Schema to Java.
csmschemagen.bat	csmschemagen	Schema generator for mapping from Java to the XML Schema.

2.5.1 csmxjc command (Binding from the XML Schema to Java)

(1) Format

```
csmxjc [option [option argument] ] ... Input schema document
```

You can omit the option. Always specify the option before the schema document.

(2) Function

Schema compiler for binding from the XML Schema to Java.

(3) Options

`-b external-binding-file`

Specifies one external binding file. For details on how to specify names of the external binding files, see the subsection *Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path*.

If this option is omitted, a java class will be generated without using external binding files.

`-d output-destination-directory`

Specifies the output destination directory of the Java source. For details on how to specify the directory name, see *Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path*.

If omitted, the current directory will become the output destination directory.

`-mark-generated`

Adds the `@javax.annotation.Generated` annotation in the generated Java source.

If omitted, the `@javax.annotation.Generated` annotation is not added in the Java source.

The format of the `@javax.annotation.Generated` annotation added in the Java source is as follows.

Different values are entered in the part indicated in *italics*, when the Java source is generated.

```
@Generated(value = "com.cosminexus.jaxb.tools.xjc.Driver", date = "yyyy-MM-ddTHH:mm:ssRFC822 timezone")
```

The following table describes the elements of the `@javax.annotation.Generated` annotation and the values set up in XML Processor:

Table 2–5: Elements of the `@javax.annotation.Generated` annotation and values set up in XML Processor

Element name	Explanation	Value
value	Completely modified class name of the schema compiler	com.cosminexus.jaxb.tools.xjc.Driver
date	Date and time when the Java source was generated from the schema	yyyy-MM-ddTHH:mm:ssRFC822 <i>timezone</i> [#]

[#]

For details on the output format, see the *Date and time pattern* of the `java.text.SimpleDateFormat` class.

(4) Input schema document

You can specify one file name for the input schema document. For details on the characters that can be specified in the file name, path delimiting characters, and how to specify a path, see *Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path*.

(5) Output Java source

Format of the output Java source

The following figure gives an overview of the format of Java source output by the schema compiler.

Figure 2–10: Overview of the format of Java source output by the schema compiler

```
//
// This file was generated by Cosminexus XML Processor 09-50
// Any modifications to this file will be lost upon recompilation of the source schema.
//

package xmlprocessor.hitachi.com;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>Java class for anonymous complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
 * :
 * <pre>
 * <?xml version='1.0' ?>
 *   <complexType>
 *     <sequence>
 *       <element name='child' type='string' minOccurs='0' maxOccurs='1' />
 *     </sequence>
 *   </complexType>
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "child"
})
@XmlRootElement(name = "root")
@Generated(value = "com.cosminexus.jaxb.tools.xjc.Driver", date = "2008-08-14T03:48:11+09:00")
public class Root {

    @XmlElement(required = true)
    @Generated(value = "com.cosminexus.jaxb.tools.xjc.Driver", date = "2008-08-14T03:48:11+09:00")
    protected String child;

    :
    :
}
```

1. Header part
2. package statement, import statement descriptor part
3. javadoc part
4. Java class body
5. Code added when specifying the -mark-generated option. Added to each generated class, field, and JavaBean property.

The details of format of the output Java source are as follows:

1. Header

Header indicating that the file was generated by Cosminexus XML Processor. The following header is inserted:

```
//
// This file was generated by Cosminexus XML Processor 09-50
// Any modifications to this file will be lost upon recompilation of the
// source schema.
//
```

2. package statement and import statement code

The package statement and import statement required in the source is generated. The contents of the generated package statement or import statement depend on the schema document entered in the schema compiler.

3. javadoc

The contents of javadoc are generated automatically for the class generated from the input schema document. The contents of javadoc are dependent on the schema document entered in the schema compiler.

4. Java class

The class generated from the input schema document is generated. The contents of the Java class depend on the schema document entered in the schema compiler.

Package and output destination of the output Java source

The package name and output destination of the output Java source is as follows:

The package name of the output Java source is determined as per the following priority order. The output destination of the Java source is determined by assuming the current directory or the directory specified in the `-d` option as the standard directory, and applying the following rules in an order:

1. When the package name is specified by custom binding

The package name specified in custom binding (`jaxb:package` element) becomes the package name of the output Java source. Convert the package name to the directory name, and create the directory name under the standard directory. Output the Java source in the created directory.

2. When the `targetNamespace` attribute is specified in the input schema document

From the target namespace described in the input schema document, generate the package name of the output Java source based on the JAXB specifications. Convert the generated package name to the directory name, and create the directory name under the standard directory. Output the Java source in the created directory.

3. When the package name is not specified by custom binding in the input schema document, and when the `targetNamespace` attribute is also not specified

The package name of Java source will be generated. Create a directory called `generated` under the standard directory, and output the Java source in the created directory.

If a Java source with the same name exists in the output destination directory, then that Java source will be overwritten.

(6) Return value

0:

Normal termination.

Value other than 0:

Abnormal termination.

(7) Notes

In Cosminexus 09-80 or later, before you execute the `csmxjc` command, set the `CSMJAXB_VM_OPTS` environment variable as follows:

```
set "CSMJAXB_VM_OPTS=--add-opens java.base/java.lang=ALL-UNNAMED --add-modules=java.activation"
```

2.5.2 csmschemagen command (Mapping from Java to the XML Schema)

(1) Format

`csmschemagen` [option [option argument]] [package-info.java file name] Java source file name

You can omit the option and `package-info.java` file. Always specify the option and `package-info.java` before the Java source file name.

(2) Function

Schema generator for mapping from Java to the XML Schema.

(3) Options

`-d output-destination-directory`

Specifies the output destination directory of the output schema document and intermediate file. For details on how to specify the output destination directory name, see *Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path*.

If omitted, the current directory will become the output destination directory.

`-encoding character-encoding`

Specifies the character encoding of the Java source to be input.

For character encoding that can be specified in Cosminexus XML Processor, see [1.3.2 Character encodings that can be processed](#).

If omitted, depending on the OS in use, the operations will differ as follows:

In Windows

The default character encoding for the OS is applied.

In UNIX

The character encoding specified in the environment variable `LANG` is applied. Even if the environment variable `LANG` is not specified, the default character encoding for the OS is applied.

(4) Input Java source

You can specify one Java source file. For details on the characters that can be specified in the file name, path delimiting characters, and how to specify a path, see *Characters that can be specified in the file name or directory name of the command, path delimiting characters, and how to specify a path*.

(5) Output schema document

Output as file name `scheman.xml` (*n* is a numeric character) in the current directory or directory specified in the `-d` option. If a file with the same name exists in the output destination directory, then that file will be overwritten.

(6) Intermediate file

- The `csmschemagen` command is used to compile the Java source file, and generate the `.class` file that is the intermediate file.
- If the Java source file contains the `package` statement, create a subdirectory under the output destination directory, and then output the `.class` file. The subdirectory name is determined based on the package name specified in the `package` statement.
- If the Java source file does not contain the `package` statement, output the `.class` file immediately under the output destination directory.
- The output destination directory of the `.class` file can be changed with the `-d` option.

(7) Return value

0:

Normal termination.

Value other than 0:

Abnormal termination.

(8) Notes

In Cosminexus 09-70 or later, when the `csmschemagen` command is executed, the `schemagen` command provided by Java SE 8 is executed.

However, use only the options supported by the `csmschemagen` command.

If invalid options are specified, the command might end with no error message displayed.

Note that if you use the command `help`, the help information for the `schemagen` command is displayed.

If you specify an uncompileable Java source file, the command ends with no error message displayed and no schema generated. Before executing the `csmschemagen` command, use the `javac` command to confirm that the target source files will not cause errors.

3

Extended Functions of Cosminexus XML Processor

This chapter describes the extended functions of Cosminexus XML Processor.

For a description of messages output in Cosminexus XML Processor, see *11. KECX (Messages Output by Cosminexus XML Processor)* in the manual *uCosminexus Application Server Messages*.

3.1 Shift_JIS Switch Function

The Shift_JIS switch function changes the character encoding applied by Cosminexus XML Processor to SJIS (x-sjis-jdk1.1.7) or MS932 (x-sjis-cp932) when the encoding attribute in the XML document specifies Shift_JIS. The following table lists the correspondence between the encoding specifications for an XML document and the applicable character encodings.

Table 3–1: Correspondence between the encoding specifications for an XML document and the applicable character encodings

Specified XML document encoding	Shift_JIS switch function setting	Applicable character encoding
Shift_JIS	Not specified	SJIS (default)
	SJIS	SJIS
	MS932	MS932
Windows-31J	--	MS932

Legend:

--: Cannot be specified

The Shift_JIS switch function is specified by setting a certain property to the instance of the XML parser or XSLT transformer. For details about how to set properties, see [4.5 How to use the properties of the Shift_JIS switch function](#).

The encoding specification using the Shift_JIS switch function is applied to the following items. The encoding cannot be applied to other items.

- encoding attribute of an input XML document
- Encoding specified for the `InputSource` object
- The output character stream when Shift_JIS is specified with the encoding attribute of the `xsl:output` element in the stylesheet or the `setOutputProperty` and `setOutputProperties` methods of the `javax.xml.transform.Transformer` class.

Care is required in the following cases:

1. When the `InputSource` object is specified as the `InputSource` for DOM and SAX parsers:
The Shift_JIS switch function is disabled if encoding other than Shift_JIS is specified for the `InputSource` object. In addition, when `Reader` is specified for the `InputSource` object, the Shift_JIS switch function is also disabled and the encoding being used by the `Reader` is applied.
2. When the `StreamSource` object is specified as the input source for the XSLT or XSLTC transformer:
When `Reader` is specified for the `StreamSource` object, the Shift_JIS switch function is disabled and the encoding being used by the `Reader` is applied.
3. When the `DOMSource` object is specified as the input source for the XSLT or XSLTC transformer:
The XML document has already undergone character code conversion when the `DOMSource` was generated. The Shift_JIS switch function is therefore disabled.
To enable the Shift_JIS switch function, you need to specify the Shift_JIS switch function in advance so the parser is used for generating the DOM node for the `DOMSource`.
4. When the `SAXSource` object is specified as the input source for the XSLT or XSLTC transformer:

If the `InputSource` object is specified for the `SAXSource` object, caution must be exercised for case 1 above. If the `XMLReader` object is specified for the `SAXSource` object, the `Shift_JIS` switch function specified in `XMLReader` is valid.

5. The JAXB does not support the `Shift_JIS` switch functionality when the `Node`, `Source`, or `Result` that is created either by the DOM parser or SAX parser where the `Shift_JIS` switch functionality is specified, is passed as an argument of the APIs of JAXB.
6. StAX does not support the `Shift_JIS` switch functionality. Therefore, for input and output of XSLT or the XSLTC transformer, if anyone of the following is used, the operation will not be guaranteed:
 - `StAXSource`
 - `StAXResult`

3.2 XSLTC Transformer Function

This section describes the XSLTC transformer function.

3.2.1 An Overview of XSLTC Transformer

The XSLTC transformer offers the same functionality as the XSLT transformer provided by Cosminexus XML Processor, with improved performance for XML document transformation.

The XSLTC transformer consists of the compiler that parses each XSLT stylesheet and the runtime processor that performs transformation. The compiler parses an XSLT stylesheet and generates the Java bytecode, called a *translet*, into memory for faster transformation. The runtime processor achieves high-speed transformation by executing the translet.

Note that the XSLTC transformer operates differently from the XSLT transformer in the following ways:

1. For the XSLTC transformer, the error check function is more simplified than that of the XSLT transformer.
2. Operations might differ between the XSLT transformer and the XSLTC transformer in the parts that are not defined in the XSLT1.0 specification.

You can expect higher transformation performance by using the XSLTC transformer instead of the XSLT transformer in the following cases:

(1) Transforming multiple XML documents by using the same XSLT stylesheet

When you transform multiple XML documents by using the same XSLT stylesheet, you can expect higher transformation performance from approach 2 (below) than approach 1, both for the XSLT and XSLTC transformers:

1. Generate the `Templates` and `Transformer` objects from the stylesheet every time and use the objects for each subsequent transformation.
2. Generate the `Templates` and `Transformer` objects from the stylesheet once only and reuse the objects for each subsequent transformation.

When using approach 2 for the XSLTC transformer, the time-consuming translet creation is needed only once. The translet is then repeatedly used to execute subsequent transformations at a faster speed. For this reason, higher performance is expected when using approach 2 for the XSLT transformer.

3.2.2 XSLTC Transformer-based System Development and Operation

The error check for the XSLTC transformer is not as strict as that for the XSLT transformer. When you develop and operate a system that uses the XSLTC transformer, perform the operation checks with the XSLT transformer as follows:

1. Make sure that error checking for the stylesheets is performed by using the XSLT transformer. This check can locate errors that cannot be detected by the XSLTC transformer, enabling you to find differences in error checks that might occur when interoperating with other systems.
2. Hitachi recommends that you verify the result of the transformation by using the XSLTC transformer is same as that by using the XSLT transformer. This validation enables you to understand whether the transformer-specific operation affects the result of the transformation and to find the differences in error checks that might occur when interoperating with other systems.

See the following cautionary notes on XSLT and XSLTC transformers:

- [6.6 General Notes on XSLT and XSLTC](#)
- [6.7 Notes on XSLT](#)
- [6.8 Notes on XSLTC](#)

3.2.3 Class Used by the XSLTC Transformer

The `TransformerFactoryXSLTC` class is used in the XSLTC transformer.

This section describes the classes used by XSLTC transformer.

(1) TransformerFactoryXSLTC class

(a) Description

This class creates the `TransformerFactory` instance for the XSLTC transformer.

(b) Package and class names

`com.cosminexus.jaxp.xsltc.TransformerFactoryXSLTC`

(c) Format

```
public class TransformerFactoryXSLTC
```

(d) List of methods

Method name	Function
<code>newInstance</code>	Creates new instances of <code>TransformerFactory</code>

(e) Method details

`newInstance` method

Description

This method creates a new `TransformerFactory` instance for the XSLTC transformer.

Format

```
public static TransformerFactory newInstance()
```

Parameters

None

Return value

This method returns the new instance of `TransformerFactory`.

Exception

None

3.2.4 How to use the XSLTC Transformer

When you use the XSLTC transformer, the `TransformerFactory` instance is generated in a different manner than the XSLT transformer. The `TransformerFactory` instance in the XSLTC transformer is generated by the following:

```
javax.xml.transform.TransformerFactory factory =  
    com.cosminexus.jaxp.xsltc.TransformerFactoryXSLTC.newInstance();
```

Other usage is the same as the XSLT transformer.

The follow shows an example of the coding:

```
import javax.xml.transform.*;  
import javax.xml.transform.stream.*;  
import com.cosminexus.jaxp.xsltc.*;  
:  
// Setting up the XSLT style sheet  
StreamSource style = new StreamSource("style.xml");  
//Generating the TransformerFactory instance  
TransformerFactory factory = TransformerFactoryXSLTC.newInstance();  
//Generating the Templates object  
Templates templates = factory.newTemplates(style);  
//Generating the XSLTC transformer  
Transformer transformer = templates.newTransformer();  
:  
//Transforming multiple XML documents using the same XSLT style sheet  
for (int i = 0; i < 10; i++) {  
    // Setting the input source  
    StreamSource source = new StreamSource(args[i]);  
    // Setting the output destination  
    StreamResult result = new StreamResult("output_" + args[i]);  
    // Execute transformation  
    transformer.transform(source, result);  
}
```

3.3 Schema cache functionality

This subsection describes the schema cache functionality.

3.3.1 Overview of the schema cache functionality

The *schema cache functionality* caches, in an object state, the syntax definitions that an XML parser acquires when parsing schema documents, and enables the re-use of the cached syntax definitions during schema validation to improve the speed of validation parsing.

With the schema cache functionality, you can create a cache in the memory and on a disk. Also, you can use a program in which APIs of JAXP1.2 are used to improve the execution speed of validation parsing without modifying the source.

This subsection describes the prerequisites and scope for using the schema cache functionality.

(1) Prerequisite conditions

The schema cache functionality is enabled only when a user program that runs on a J2EE server uses the `javax.xml.parsers.DocumentBuilder` class (DOM parse) to parse XML documents.

The following are the restrictions when using the schema cache functionality:

- The schema cache functionality is not applicable to the schema validation using the `javax.xml.validation` package.
- When passing the `File` object to an argument of the `parse` method, some part of the name of the schema document file that is maintained within the parse will not be in conformity with the name of the file that is created while setting up a cache. Therefore, you cannot use the schema cache functionality.

(2) Scope

The following schema documents will be cached:

- Schema documents specified in the schema definition file
- Schema documents are referenced directly or indirectly from the schema documents that are defined in the schema definition file, with the following elements:
 - `import` element
 - `include` element
 - `redefine` element

Note that when starting multiple J2EE servers on one machine, the schema cache functionality operates independently on every J2EE server.

3.3.2 Processing of the schema cache functionality

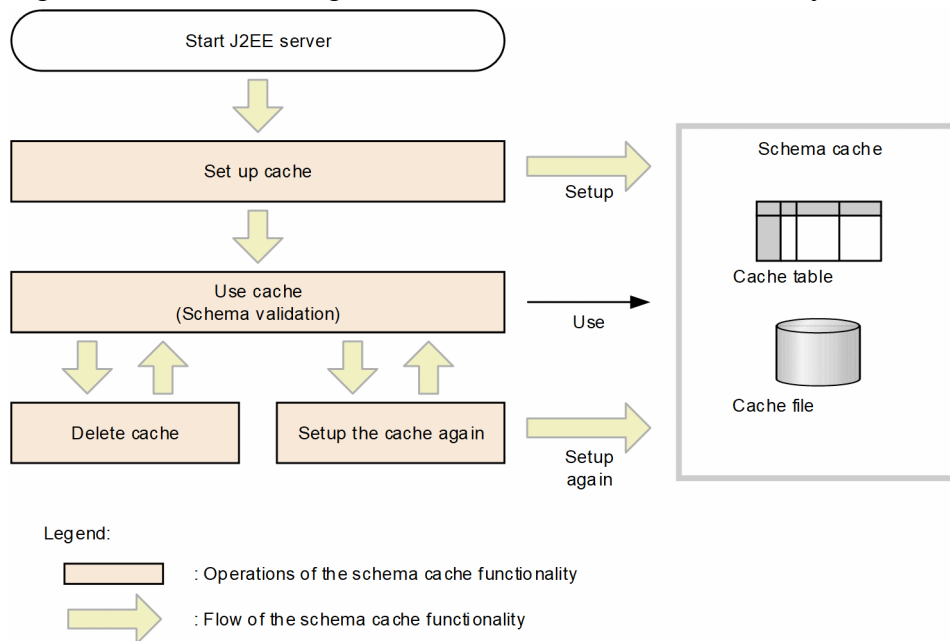
The schema cache functionality includes the following for operating a schema cache:

- Setting up a schema cache

- Using the schema cache
- Deleting and resetting up the schema cache

The following figure shows the processing of the schema cache functionality.

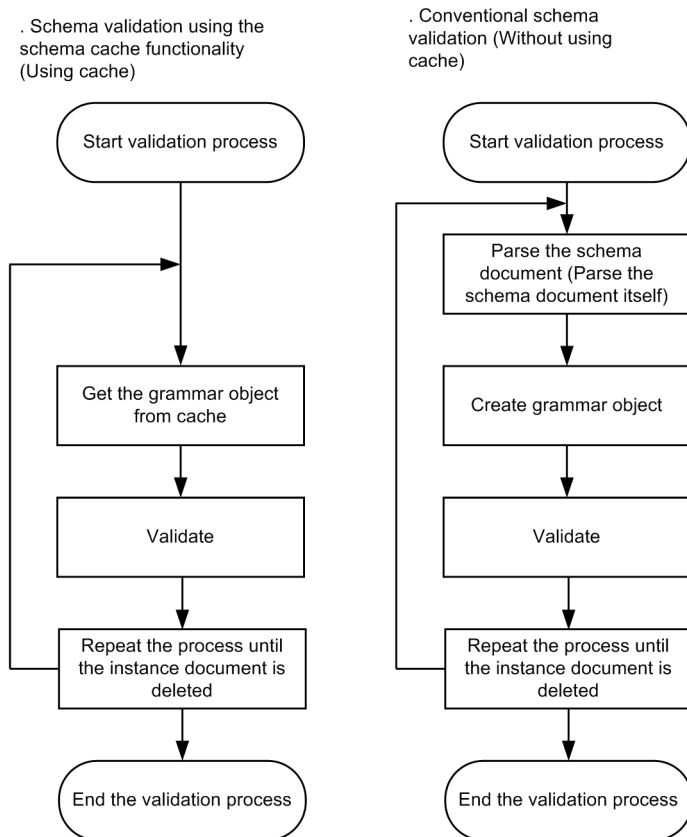
Figure 3–1: Processing of the schema cache functionality



Conventional schema validation parses a schema document during validation and creates grammar objects even when the same schema is used for each validation; however, if you use the schema cache functionality, already existing grammar objects are reused to reduce the overhead of schema document parsing.

The following figure shows an example where one schema document is used to validate multiple instance documents. The example shows the differences in the process when using the schema cache functionality with a cache and when using the conventional process scheme validation without a cache:

Figure 3–2: Difference in the process of schema validation using the schema cache functionality and the process of past schema validation



3.3.3 Setting up the schema cache

To use the schema cache functionality, you must decide a schema document for cache in advance and set up a schema cache. Therefore, set up the following property file appropriately:

- Schema definition file
- User-defined file (`usrconf.properties`) of the J2EE server

The above mentioned file corresponds to the Java property file. Therefore, for basic syntax rules, follow the rules used in the Java property file. Note that for details about the syntax rules of the property files, see *Javadoc* of the `java.util.Properties` class.

This subsection described the property files.

(1) Schema definition file

You can define the following items in a schema definition file:

- Schema document to cache
- Cache format
- Location on a disk in which the schema cache is saved

You can specify any file name and save location for the schema definition file.

The following schema documents will be cached:

- Schema documents directly specified in the schema definition file
- Schema documents that are referenced directly or indirectly from the schema documents that are defined directly in the schema definition file, with the following elements:
 - `import` element
 - `include` element
 - `redefine` element

The following table lists the properties that you can code in a schema definition file. If you specify any other property, that property will be ignored.

Table 3–2: Properties that can be coded in a schema definition file

No.	Property name	Specified value
1	<code>schema_path</code>	Specifies an absolute path that indicates a directory where the schema document exists.
2	<code>diskcache_path</code>	Specifies an absolute path that indicates a directory where the disk cache is saved.
3	<code>schema_xxx</code> (xxx is any character string that has more than one character)	<p>Specifies a relative path that indicates a file of the schema document to be cached and the disk cache specifier. You can omit the disk cache specifier.</p> <p>For the disk cache specifier, code <code>D</code> at the end of the relative path of the schema document. The schema document cache specified by the disk cache specifier is created on the disk.</p> <p>If there is no coding of the disk cache specifier, a cache is created in the memory.</p> <p>A space or a tab that is connected to the file name and the disk cache will be ignored.</p>

Use a forward slash (/) as a delimiter in a path specified for each property, in Windows and UNIX. Using a forward slash (/) at the end of an absolute path for the directory name is optional.

If the same property name is specified more than once, the last specified value of the property will be valid.

The following are the examples of schema definition files:

In Windows

```
schema_path=C:/usr/app/xmlparser/schemafiles
diskcache_path=C:/usr/app/xmlparser
schema_01=schema/type1.xsd
schema_02=schema/type2.xsd,D
```

In UNIX

```
schema_path=/usr/app/xmlparser/schemafiles
diskcache_path=/usr/app/xmlparser
schema_01=schema/type1.xsd
schema_02=schema/type2.xsd,D
```

When `schema_01=schema/type1.xsd` is specified, the disk cache specifier has not been coded, so cache of the schema document will be performed in the memory.

When `schema_02=schema/type2.xsd,D` is specified, the disk cache specifier has been coded, and therefore, the schema document is cached on the disk.

The following are the operations for specifying the schema definition files:

- You must specify the `schema_path` property and the `diskcache_path` property.
If any one of the above two properties is not specified, a warning message (KECX09503-W) indicating that the required property is not specified will output to an error file, and schema validation is executed without using the schema cache.
- Even if the same directory is specified in the `diskcache_path` property on multiple J2EE servers, the disk cache is managed independently on each server.
A directory having the name same as that of the J2EE server is created in the directory specified in the `diskcache_path` property, and a cache file is created in the directory. When the disk cache is deleted, the cache file for each created subdirectory will be deleted.
- When the specified of the `diskcache_path` property is not an absolute path, the warning message (KECX09507-W) is output to an error file, and schema validation is executed without using the schema cache.
- When you cannot access to a directory that is specified in the `diskcache_path` property, the warning message (KECX09505-W) indicating that the directory for saving the disk cache cannot be accessed will output to an error file, and schema validation is executed without using the schema cache.
- When the specified value of the `schema_path` property is not an absolute path, the warning message (KECX09508-W) is output to the error file, and schema validation is executed without using the schema cache.
- When the specified value of the `schema_XXX` property is not a relative path, the warning message (KECX09509-W) is output to an error file, and the specification of the `schema_XXX` property will be ignored.
- When an attempt to write grammar objects in a file fails while creating a disk cache, the warning message (KECX09511-W) is output to an error file. A parser validates the schema using the schema document and not the cache for which an attempt to write the grammar objects has failed.
- When the same schema document is specified with multiple `schema_XXX` properties, only one cache corresponding to the schema document is created. In such cases, if the disk cache specifier is specified to any of the properties among all the properties, a cache will be created on the disk. If the disk cache specifier is not specified to any of the properties, the cache will be created in the memory.

For details about the precautions related to the schema definition files, see the subsection [6.18.3 Precautions related to schema definition files](#).

(2) J2EE server user-defined file

The user-defined file (`usrconf.properties`) of the J2EE server defines a class, to be initialized when the J2EE server starts, and a system property to be set up.

For details about the user-defined files of J2EE servers, see *2. Files Used in J2EE Servers* in the *uCosminexus Application Server Definition Reference Guide*.

The following table lists the properties coded in the user-defined file of the J2EE server, when the schema cache functionality is used.

Table 3–3: Properties coded in the user-defined file of the J2EE server

No.	Property name	Specified value
1	<code>ejbserver.application.InitTermProcessClasses</code>	Specifies the initialization and termination class of the J2EE extended container. Specify <code>com.cosminexus.jaxp.impl.parsers.util.XMLGrammarCaching</code> that is required to set up a schema cache, using the schema cache functionality.

No.	Property name	Specified value
2	<code>com.cosminexus.jaxp.grammar_caching.preload</code>	Specifies whether to use the schema cache functionality or not. You can specify either ON or OFF. The default value is OFF. If you specify ON, the schema cache functionality can be used. The XML parser executes the schema validation using the schema cache. If you specify OFF, the schema cache functionality cannot be used. The XML parser executes the schema validation without using the schema cache. Note that this property controls all the schema cache functionality. You must specify ON to use the schema cache functionality for J2EE server. If you specify OFF, the schema cache functionality will not be enabled after the J2EE server starts.
3	<code>com.cosminexus.jaxp.grammar_caching.config</code>	Specifies an absolute path of the schema definition file.

You can use the schema cache functionality only when corresponding to all the following conditions:

- Specifies the `ejbserver.application.InitTermProcessClasses` property.
- Specifies ON in the `com.cosminexus.jaxp.grammar_caching.preload` property.
- Specifies the `com.cosminexus.jaxp.grammar_caching.config` property.

The following is an example of the user-defined files for J2EE servers:

```
# Specifies the class to be initialized when Cosminexus starts
ejbserver.application.InitTermProcessClasses=com.cosminexus.jaxp.impl.parser
s.util.XMLGrammarCaching
# Enables the cache functionality
com.cosminexus.jaxp.grammar_caching.preload=ON
# Absolute path of schema definition file defined in the schema document to
be maintained in the cache
com.cosminexus.jaxp.grammar_caching.config=C:/usr/app/xmlparser/config.prope
rties
```

The following are the operations for specifying the user-defined files for J2EE servers:

- When the following value is specified in the `com.cosminexus.jaxp.grammar_caching.preload` property, OFF must be specified:
 - A value other than ON or OFF is specified
 - A property is not specified
- When you cannot access the schema definition file specified in the `com.cosminexus.jaxp.grammar_caching.config` property, even if ON is specified in the `com.cosminexus.jaxp.grammar_caching.preload` property, you cannot use the schema cache functionality (warning message KECX09504-W)

3.3.4 Using the schema cache

When using the XML parser to execute the schema validation, the operations of the user for using the schema cache functionality are not required. The XML parser decides whether to use the schema cache functionality or not, and when using the schema cache functionality, the schema validation is executed automatically using the schema cache. When the

schema cache is not used or when the cache is not found, schema validation is executed after creating grammar objects same as is done in the past XML parser.

Note that when a failure occurs while reading the grammar objects from the disk cache file, a warning message (KECX09512-W) is output. The XML parser does not use a cache that could not be read, so the schema validation is executed using the schema document.

With the schema cache functionality, you can use the following commands to control a cache or to check status:

Table 3–4: Schema cache functionality commands

No.	File name of the command		Functionality
	Windows	UNIX	
1	cacheoff.bat	cacheoff	Disables the schema cache functionality, and deletes the cache.
2	cacheon.bat	cacheon	Enables the schema cache functionality, and reset up the cache.
3	cachestate.bat	cachestate	Displays status of the schema cache functionality.

All the command files are saved in the `bin` directory below `JAXP_DIR`[#]. To use the commands, you must set up the `bin` directory below `JAXP_DIR`[#] with the `PATH` environment variable.

#

This is the `jaxp` directory in the directory where Cosminexus is installed.

For details about the commands, see the subsection [3.3.5 Deleting and resetting the schema cache](#).

3.3.5 Deleting and resetting the schema cache

For deleting or changing schema documents to be cached, you must use commands to delete or reset up the cache. You can use the commands to apply the changes of the schema document in the cache without stopping the J2EE server.

For details about the precautions to be taken when deleting and resetting up the cache, see the subsection [6.18.4 Precautions related to reset up and deletion of a cache](#).

This subsection describes the commands to be executed while deleting and resetting up the cache:

(1) cacheoff command (deleting cache with XML parser)

(a) Format

```
cacheoff J2EE server name
```

(b) Operation while executing the cacheoff command

The following are the operations when using the `cacheoff` command. Note that if the schema cache functionality is disabled, no operation will be performed even when using the `cacheoff` command.

- When you execute the `cacheoff` command, the XML parser releases all the cache from the memory, and deletes the directories having the same name as the J2EE server in the directory specified in `diskcache_path` of the schema definition file, and all the disk cache created below the directory.

- The schema cache functionality is disabled after the `cacheoff` command is executed, and the XML parser executes the schema validation without using the cache.
- The `cacheoff` command instructs the deletion of the cache in the XML parser, but does not directly delete the cache. The cache is deleted, when the DOM parse is passed for the first time, after the `cacheoff` command is executed.
- When using the `cacheoff` command, the directory "directory or server name specified in `diskcache_path`" is displayed, and the user is requested to confirm the deletion. Even when the cache is not created on the disk, and only the directory to save the disk cache is to be created, the user will be requested to confirm the deletion.
- An error message (KECX09506-E) is output in the standard error output, when using the `cacheoff` command, and an internal directory and the internal files are not accessible due to the abnormalities in the system, and the command process is interrupted.
- The error message (KECX09510-E) is output, when you specify the J2EE server name that does not exist in the argument of the `cacheoff` command.
- The error message (KECX09510-E) is output when ON is not specified in the `com.cosminexus.jaxp.grammar_caching.preload` property of the user-defined file of the J2EE server, and the `cacheoff` command is used for the J2EE server.
- When there is a mistake in the argument of the command, the following error message is output and the process is interrupted:

```
KECX09513-E cacheoff : Invalid parameter.
usage : cacheoff J2EE_server_name
```

(2) cacheon command (cache resetting with XML parser)

(a) Format

`cacheon J2EE server name`

(b) Operation when executing the cacheon command

The following are the operations, when using the `cacheon` command:

- When using the `cacheon` command, the XML parser releases the cache from the memory and deletes the directories having the name same as that of the J2EE server in the directory specified in `diskcache_path` of the schema definition file, and all the disk cache created below the directory. Consequently, the cache is set up again according to the contents of the schema definition file.
- The schema cache functionality is enabled after using the `cacheon` command, and the XML parser executes the schema validation using the cache.
- The `cacheon` command instructs the reset up of cache with the XML parser, but does not directly delete or reset up the cache. The cache is deleted or reset up when the XML parse is passed for the first time, after using the `cacheon` command.
- When using the `cacheon` command, an internal directory and the internal files are not accessible because of the abnormalities in the system, the error message (KECX09506-E) is output in the standard error output, and the command process is interrupted.
- When you specify the J2EE server name that does not exist in the argument of the `cacheon` command, the error message (KECX09510-E) is output.

- When ON is not specified in the `com.cosminexus.jaxp.grammar_caching.preload` property of the user-defined file of the J2EE server, and the `cacheon` command is executed for that J2EE server, the error message (KECX09510-E) is output.
- When there is a mistake in the argument of the command, the following error message is output and the process is interrupted:

```
KECX09513-E cacheon : Invalid parameter.
usage : cacheon J2EE_server_name
```

(3) cachestate command (display the cache functionality status)

(a) Format

```
cachestate [-d] [J2EE server name]
```

(b) Operation when executing the cachestate command

The following are the operations when using the `cachestate` command:

- If the `cachestate` command is executed without the `-d` option, the status of the schema cache functionality is output in the standard output. The following table describes the displayed contents:

Table 3–5: Status of the schema cache functionality to be displayed

No.	Status of schema cache functionality	Content displayed
1	Enabled	"KECX09501-I J2EE Server 'J2EE server name': schema cache: ON"
2	Disabled	"KECX09501-I J2EE Server 'J2EE server name': schema cache: OFF"

- When the `cachestate` command is executed, and an internal directory and internal files are not accessible because of the abnormalities in the system, the error message (KECX09506-E) is output in the standard error output, and the command process is interrupted.
- When you specify a J2EE server name that does not exist in the argument of the `cachestate` command, the error message (KECX09510-E) is output.
- When J2EE server name of the argument of the `cachestate` command is omitted, the information of all the J2EE servers is output.
- In the user-defined file of the J2EE server, information is output, if ON is not specified in the `com.cosminexus.jaxp.grammar_caching.preload` property, no information will be output, even when you specify the `-d` option in the `cachestate` command and execute the command.
- When there is a mistake in the argument of the command, the following error message is output, and the process is interrupted:

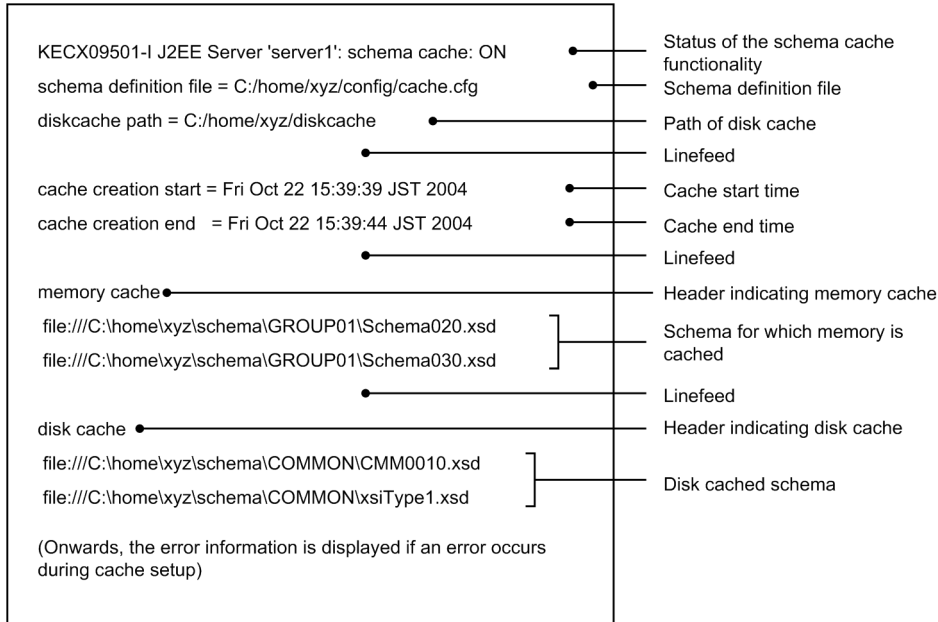
```
KECX09513-E cachestate : Invalid parameter.
usage : cachestate [-d] [J2EE_server_name]
```

- If the `-d` option is specified in the `cachestate` command and the command is executed, the following information is displayed in the standard output:
 - Schema cache functionality status (ON or OFF)
 - The current valid schema definition file

- The current valid `diskcache_path` (directory where the disk cache is created)
- Start and end time of the cache setup
- Cache setup status of the schema document
- Error information during the cache setup

The following figure shows the output format of the information.

Figure 3–3: Output format when executing the `cachestate` command



When you do not specify the server name, only the information about the number of servers will output. Even if the error KECX09506-E occurs during the execution, the information of the remaining servers will output. The content of this information is cleared whenever the cache is reset and updated.

(4) Relation of the synchronous processing of each command and parsing

The following table describes the relation of the synchronous processing of the `cacheon` command, `cacheoff` command, `cachestate` command, and the parsing.

Table 3–6: Relation of the synchronous processing of each command and parsing

No.	Action to be executed	System status		
		Other than when parsing and the cases mentioned at the right side	When setting up cache	When deleting cache
1	Parsing	Start parsing (start the process immediately).	Start parsing after the setup is complete.	Start parsing after completing the deletion.
2	<code>cacheon</code> command	Executed in the following order: 1. Immediately end the command. However, the schema cache setup is not executed at that time. 2. Start the next parsing. 3. Setup cache.		
3	<code>cacheoff</code> command	Execute in the following order: 1. Immediately end the command. However, schema cache is not deleted at that time. 2. Start the next parse.		

No.	Action to be executed	System status		
		Other than when parsing and the cases mentioned at the right side	When setting up cache	When deleting cache
		3. Delete cache.		
4	cachestate command	Display the following information: When the schema cache functionality is enabled: <pre>"KECX09501-I J2EE Server 'J2EE server name': schema cache: ON"</pre> When the schema cache functionality is disabled: <pre>"KECX09501-I J2EE Server 'J2EE server name': schema cache: OFF"</pre>	"KECX09501-I J2EE Server 'J2EE server name': schema cache: displays OFF".	

3.3.6 About using the most appropriate schema cache

When you use the schema cache functionality, you can reduce the parsing time for validation. However, the problems such as increase in the memory usage or when the disk cache is used, the expected effects might not output.

This subsection describes how to avoid such problems and use the appropriate schema cache.

(1) Relation of the size and the validation time of schema and instance document

A schema document is coded with XML. The parsing is performed before the validation check, and converted to internal grammar objects. Using cache, you can omit the processing of setting the grammar objects from the schema document. The following time is required for parsing the validation check:

validation-time-of-the-instance-document + time-for-setting-grammar-objects-from-schema-document

The time required for setting grammar objects can be omitted from the cache, so the effect of the schema cache as displayed in the first condition will be more in the following conditions:

1. *size-of-the-schema-document > size-of-the-instance-document*
2. *size-of-the-schema-document < size-of-the-instance-document*

(2) Difference between memory cache and disk cache

When using the schema cache functionality, grammar objects are set up from the schema document in advance, so you can omit the time required for setting the grammar objects during the validation.

However, when the memory and the disk cache are used, the time required for parsing the validation check is as follows:

- Memory cache
For parsing the validation check, the time taken for validating the instance document is required.
- Disk cache

The following time is required for parsing the validation check:

validation-time-of-the-instance-document + *deserialize[#]-time* + *I/O-time*

#

The processing that converts the data in such a way (serialized) so that the file with all the Java objects can be saved with the format of the original object.

(3) Precautions when using the memory cache

When using the memory cache, after a cache is setup, the cache remains in the memory. Therefore, adjust the memory usage within the permitted range of resources. It is effective to select a schema document that is frequently used, and then using the memory cache with the priority.

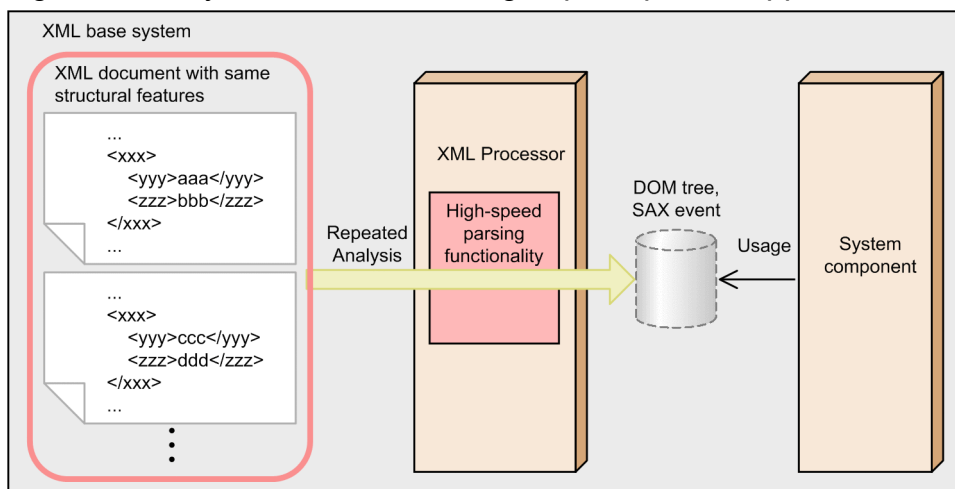
3.4 High-speed parse support function

This section describes the high-speed parse support function.

3.4.1 Overview of the high-speed parse support function

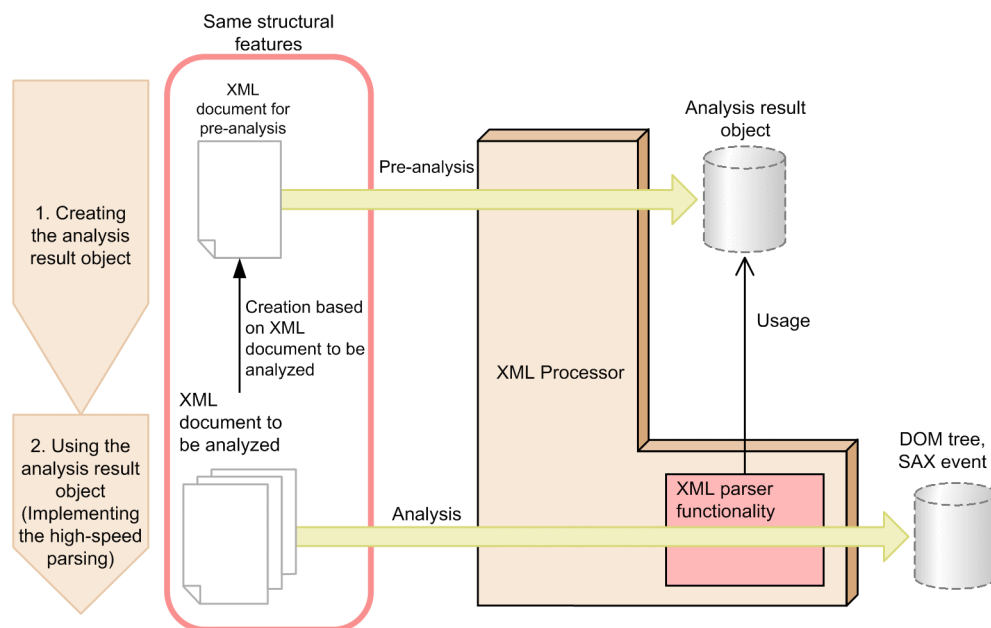
The *high-speed parse support function* improves the execution speed of the parsing by studying the features of the XML document to be parsed through pre-parsing, and thereafter using the study results when parsing the XML document. The high-speed parse support function is suitable for systems that perform iterative parsing of XML documents in which the structural features such as order of elements and attributes or embedded relations are same. The following figure shows a system in which the high-speed parse support function is used:

Figure 3–4: System in which the high-speed parse support function is used



The features of the XML document studied through pre-parsing are recorded in the *prepared object*. The following figure shows the flow of generating the prepared object and implementing high-speed parse.

Figure 3–5: Flow of generating the prepared object and implementing high-speed parse



1. Generating the prepared object

The Prepared object is generated by performing preparse of information, such as the order of elements, the order of attributes, and the iteration of elements from the Pre-Parse XML document, in Cosminexus XML Processor. The *Pre-Parse XML document* is an XML document with the same structural features as the XML document for parsing. The Pre-Parse XML document must be created by the user. As an XML document generally has a free structure, a prepared object corresponding to all XML documents cannot be generated. Therefore, the Pre-Parse XML document is created by concentrating on the document structure with high occurrence frequency, and after considering the XML documents for parsing.

2. Using the prepared object (implementing high-speed parse)

The XML document is parsed using the prepared object when parsing with the XML parser function of Cosminexus XML Processor in which the prepared object is set up. The parsing speed will improve if the structure of the XML document for parsing matches with the structure of the document recorded in the prepared object.

When using the high-speed parse support function as per the flow shown in Figure 3-5, you must describe the codes for generating the prepared object and setting it up in an XML parser, in the user program. For details about how to use the high-speed parse support function by describing the codes in the user program, see [3.4.2 Flow of operations for high-speed parse](#).

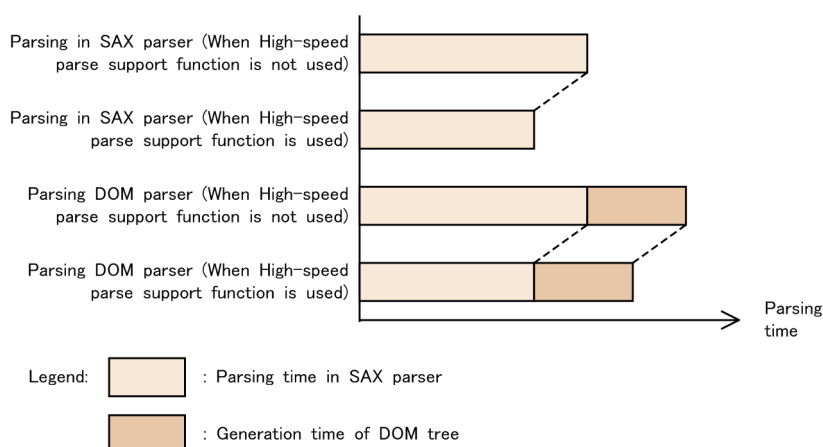
However, Hitachi does not guarantee that the parse speed can be improved than that of the normal speed in all cases. Therefore, when using the high-speed parse support function, Hitachi recommends that you evaluate the performance by actually using the XML document for parsing.

When using the high speed parse support function, check the contents of [6.19 Notes on the high-speed parse support function](#) as well.

Reference note

About the amount of the parsing time that can be reduced using the high-speed parse support function

The high-speed parse support function reduces the parsing time in the SAX parser. Furthermore, as parsing with a DOM parser is implemented by parsing with the SAX parser and generation of a DOM tree, the parsing time of the DOM parser can also be reduced. However, the time for generating the DOM tree does not change even after using the high-speed parse support function. Therefore, as shown in the following figure, in comparison to parse with SAX parser, the amount of the reduced time as against the entire parsing time is less.



Also, when the validation check is performed, the amount of parsing time for the validation check process is high, and therefore, the amount of the reduced parsing time will be relatively less. Particularly, when the

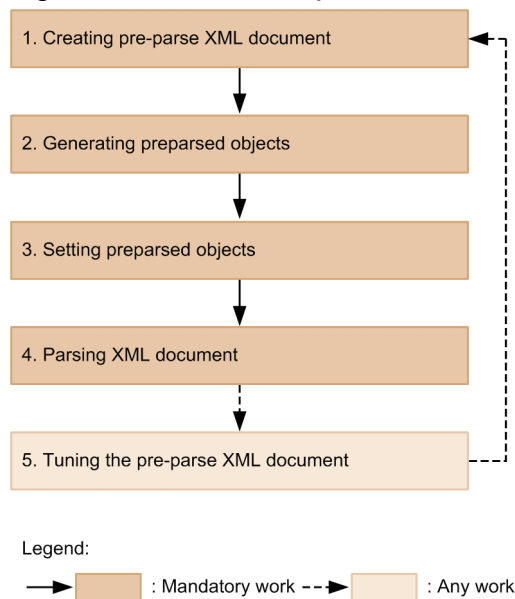
validation process is executed without using the `javax.xml.validation.Schema` class, the parsing time is hardly reduced.

Even for parsing a small-size XML document, the amount of initialization time of the XML parser in the entire parsing time is high, and therefore, the amount of the reduced parsing time will be relatively lesser.

3.4.2 Flow of operations for high-speed parse

This subsection describes the flow of operations executed by the users when the high-speed parse support function is executed by describing the code in the user program.

Figure 3–6: Flow of operations for using the high-speed parse support function



Each operation is as follows:

1. Creating a Pre-Parse XML document

Create the Pre-Parse XML document considering the structural features of the XML document for parsing. For details about the policies for creating the Pre-Parse XML document, see [3.4.3 Creating the Pre-Parse XML document](#).

2. Generating a preparsed object

Parse the Pre-Parse XML document, and then generate the preparsed object. For details about how to generate the preparsed object, see [3.4.4 Generating the preparsed object](#).

3. Setting up the preparsed object

Set up the preparsed object in an XML parser of Cosminexus XML Processor. For details about how to set up the preparsed object, see [3.4.5 Setting up the preparsed object](#).

4. Parsing the XML document

Parse the XML document by the `parse` method of the XML parser in which the preparsed object is set up. For details about the `parse` methods for high-speed parse support function, see [3.4.6 Parsing the XML document](#).

5. Tuning the Pre-Parse XML document

Tune the Pre-Parse XML document as and when required. The parsing speed can be improved furthermore by optimizing the Pre-Parse XML document after examining the tuning information. For details about how to tune the Pre-Parse XML document, see [3.4.9 Tuning the Pre-Parse XML document](#).

For details about coding examples to be described in the user program for using the high-speed parse support function, see [3.4.8 Coding example for using the high-speed parse support function](#).

3.4.3 Creating the Pre-Parse XML document

Create the Pre-Parse XML document considering the structural features of the XML document for parsing. Create a Pre-Parse XML document as per the following policies:

Table 3–7: Policies for creating a Pre-Parse XML document

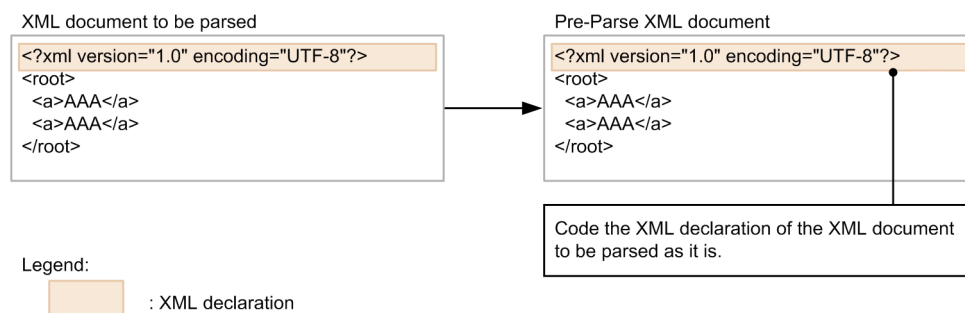
No.	Applicable location	Policy
1	<i>XML declaration</i>	Must match with the XML document for parsing.
2	<i>Elements and attributes</i>	Describe the elements and attributes described in the XML document for parsing.
3	<i>Order of elements and attributes</i>	Must match with the XML document for parsing.
4	<i>Embedded structure of elements</i>	Must match with the XML document for parsing.
5	<i>Iteration structure of elements</i>	Code the iterative elements in an iterative format.
6	<i>Text, CDATA sections, and attribute values</i>	Need not match with the XML document for parsing.
7	<i>DTDs, comments, and processing instructions</i>	Need not be coded.

Each of these policies is described as follows by citing specific coding example:

(1) XML declaration

The XML declaration of the XML document for parsing is coded as it is in the Pre-Parse XML document. The following figure shows an example:

Figure 3–7: Example for describing XML declaration (policy for creating a Pre-Parse XML document)

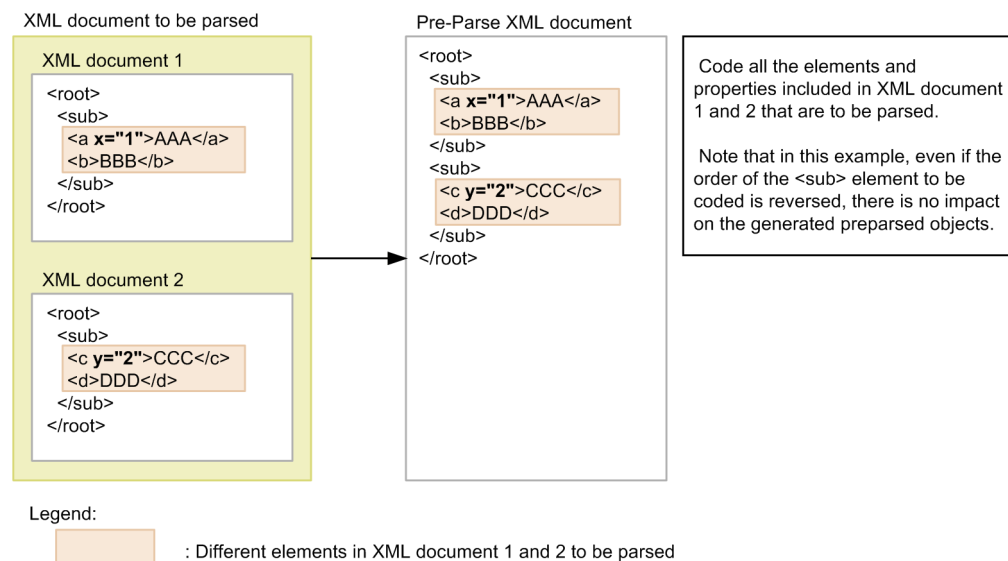


(2) Elements and attributes

The elements and attributes of the XML document for parsing are coded in the Pre-Parse XML document. The information about the elements and attributes described in the Pre-Parse XML document is recorded in the preparsed object that is generated.

If there are different elements and attributes in the XML documents for parsing, then code all those elements and attributes in the Pre-Parse XML document. The following figure shows an example:

Figure 3–8: Example of coding of elements and attributes (policy for creating a Pre-Parse XML document)



However, the text in the elements, CDATA sections, and the contents of the attribute values need not match the XML document for parsing. For details, see [3.4.3 \(6\) Text, CDATA sections, and attribute values](#).

The following codes are not differentiated according to XML standards, but are differentiated when pre-parsing as different elements:

- Blank elements and blank element tags (example: `<a>` and `<a/>`)
- Elements with different number and position of linefeed characters, space characters, or tags

If these elements exist in the XML document for parsing, they must be differentiated and coded in the Pre-Parse XML document. The following figure shows an example:

Figure 3–9: Example of coding of blank elements and blank element tags (policy for creating the Pre-Parse XML document)

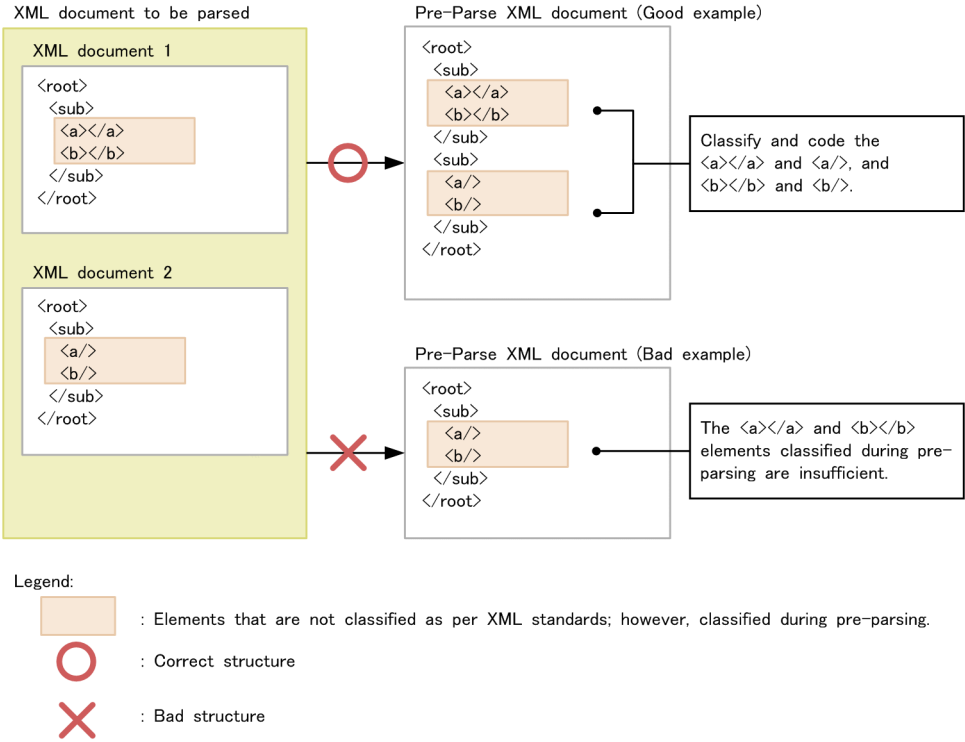
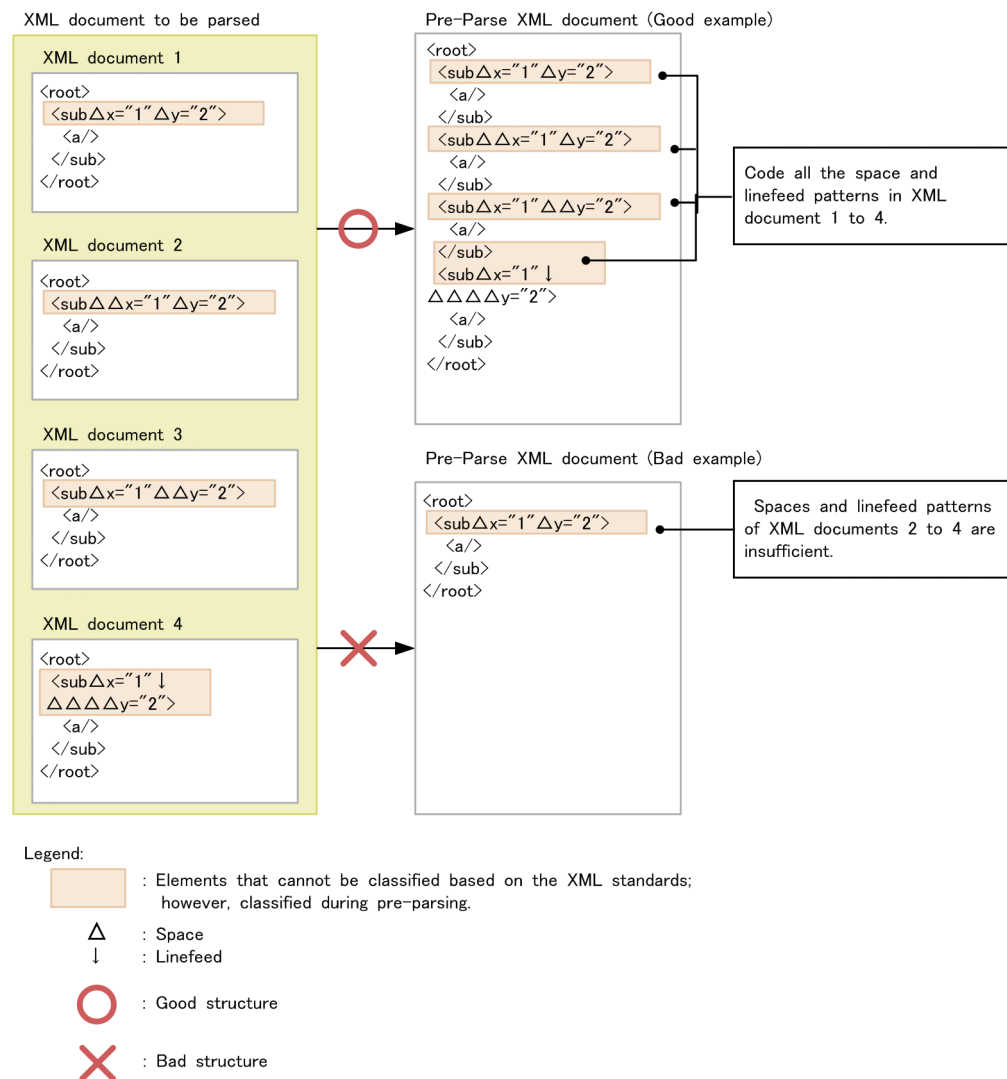


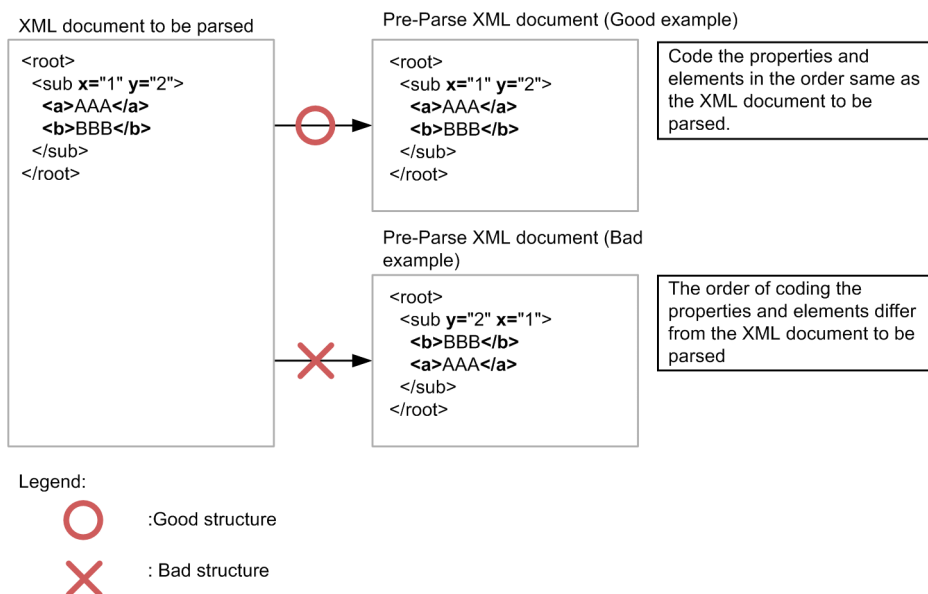
Figure 3–10: Example of coding elements with different number and position of linefeed characters or blank characters (policy for creating the Pre-Parse XML document)



(3) Order of elements and attributes

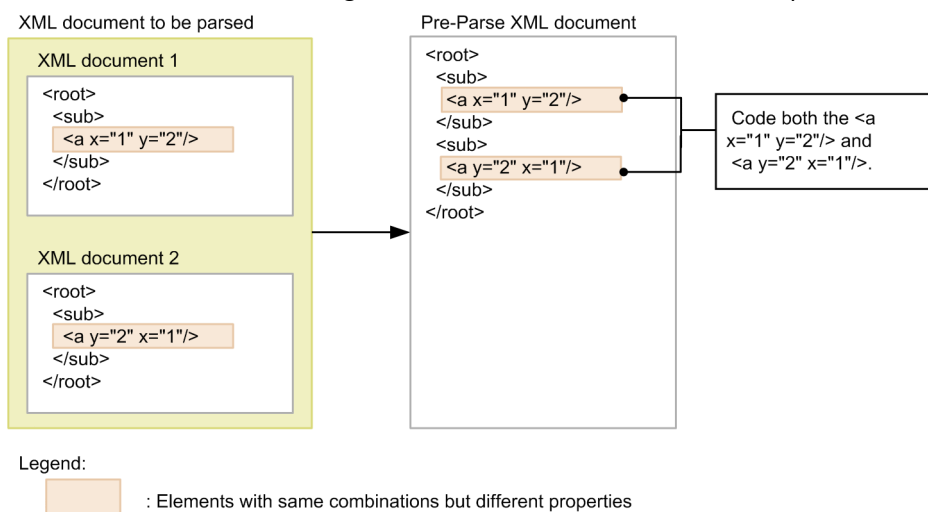
The elements and attributes of the XML document for parsing are coded in the same order in the Pre-Parse XML document. The following figure shows an example:

Figure 3–11: Example of describing elements and attributes (policy for creating the Pre-Parse XML document, and order of elements and attributes)



If elements and attributes with the same combination but different order are described in the XML document for parsing, all such elements and attributes will be described in the Pre-Parse XML document. The following figure shows an example:

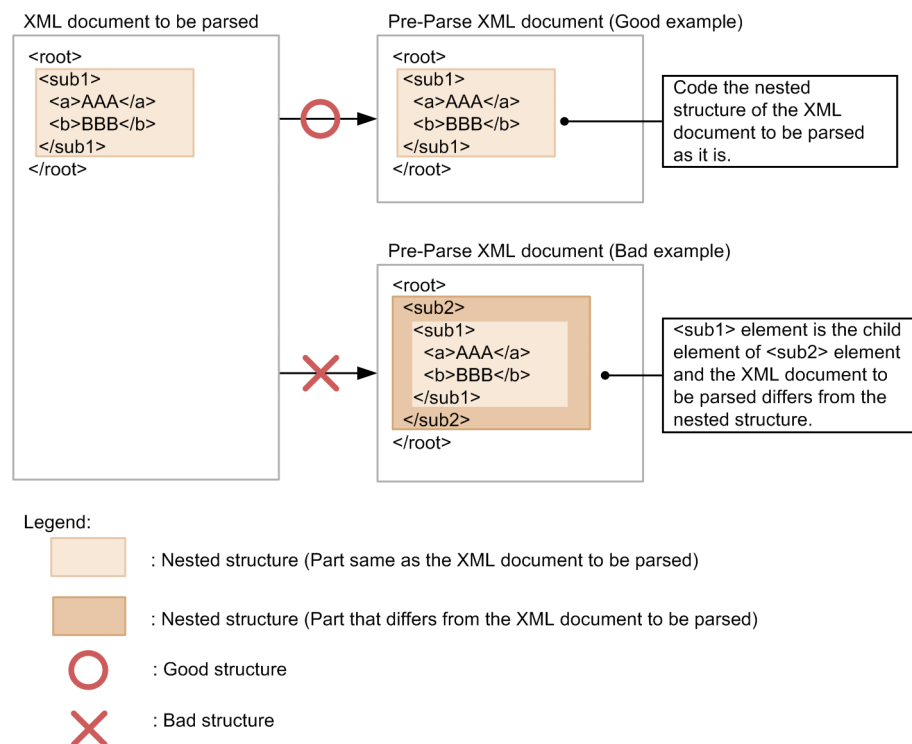
Figure 3–12: Example of describing attributes with the same combination but different order (policy for creating the Pre-Parse XML document)



(4) Embedded structure of elements

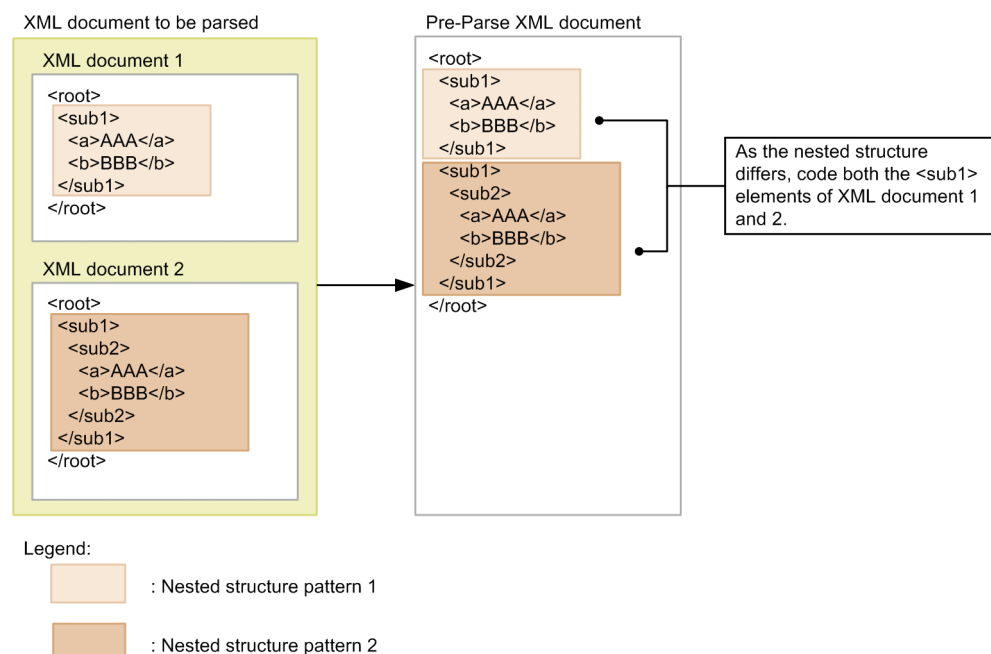
The embedded structure of elements in the XML document for parsing is described in the same format in the Pre-Parse XML document. The following figure shows an example:

Figure 3–13: Example of describing the embedded structure of elements (Policy for creating the Pre-Parse XML document)



If there are many different embedded structures in the XML document for parsing, all embedded structures will be coded in the Pre-Parse XML document. The following figure shows an example:

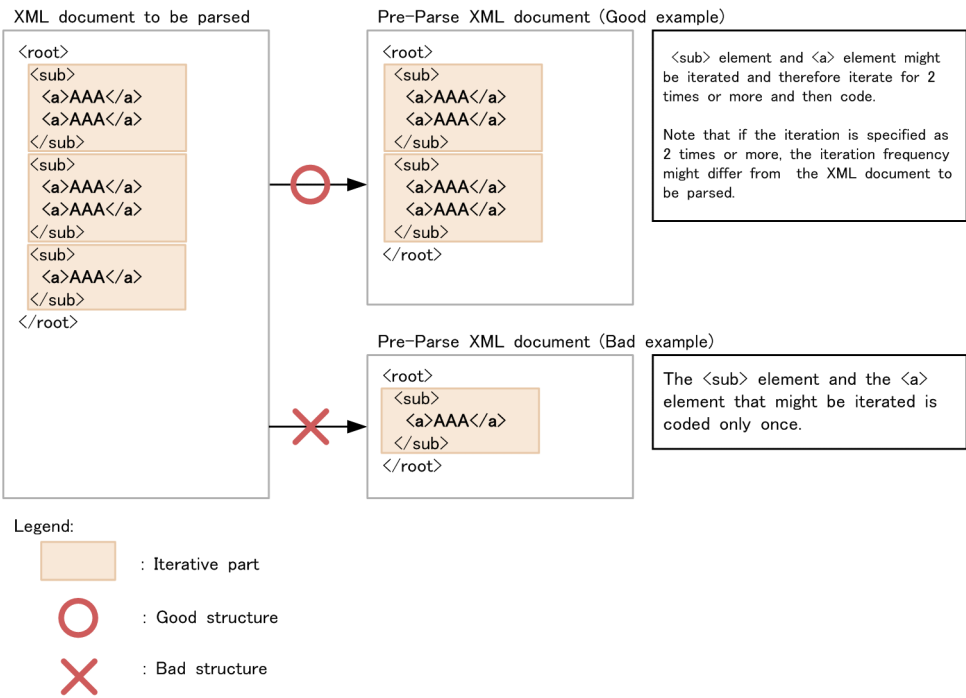
Figure 3–14: Example of coding multiple embedded structures of elements (policy for creating the Pre-Parse XML document)



(5) Iteration structure of elements

If the XML document for parsing contains iterative elements, the iterative elements are described in continuation for two or more times in the Pre-Parse XML document. The elements described continuously for two or more times are treated as iterative elements when parsing. If the frequency of continuous description in the pre-parsing XML document is two or more times, there would be no impact on the performance of the high-speed parse support function even if the iteration frequency differs from that of the XML document for parsing. The following figure shows an example:

Figure 3–15: Example of coding the iteration structure of elements (Policy for creating the Pre-Parse XML document)

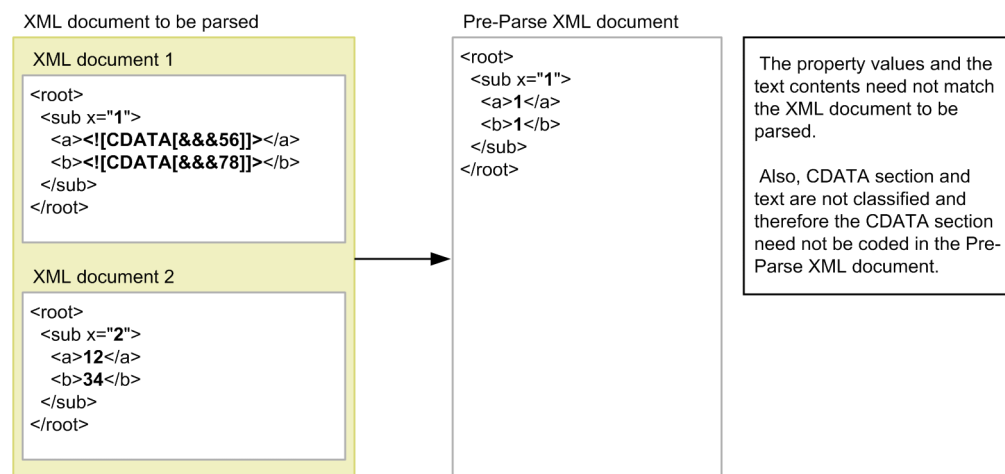


(6) Text, CDATA sections, and attribute values

The text, CDATA sections, and contents of the attribute values described in the Pre-Parse XML document do not have an impact on the preparsed object that is generated. Therefore, the text, CDATA sections, and contents of the attribute values in the Pre-Parse XML document need not match with the XML document for parsing.

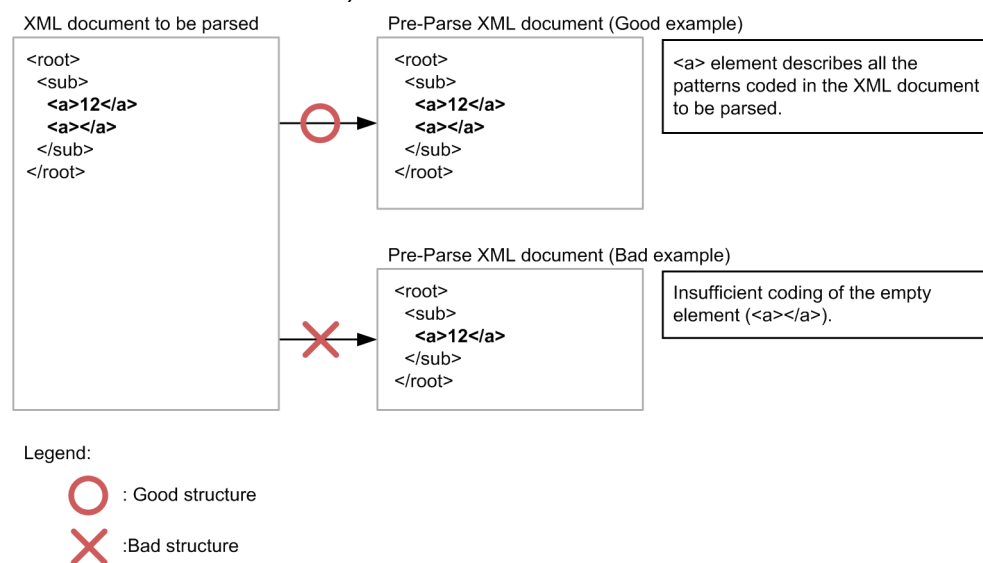
Also, the text and CDATA sections are not differentiated (the preparsed object is same no matter which one is coded). The following figure shows an example:

Figure 3–16: Example of coding the text, CDATA sections, and attribute values (policy for creating the Pre-Parse XML document)



Elements with contents (example: `<a>12`) and empty elements (example: `<a>`) are differentiated. Furthermore, as the linefeed, tab, and space are considered as text, the elements containing such text (example: `<a>[linefeed]`) and empty elements (example: `<a>`) are differentiated. Therefore, if the XML document for parsing contains such elements, all elements must be differentiated and described in the Pre-Parse XML document. The following figure shows an example:

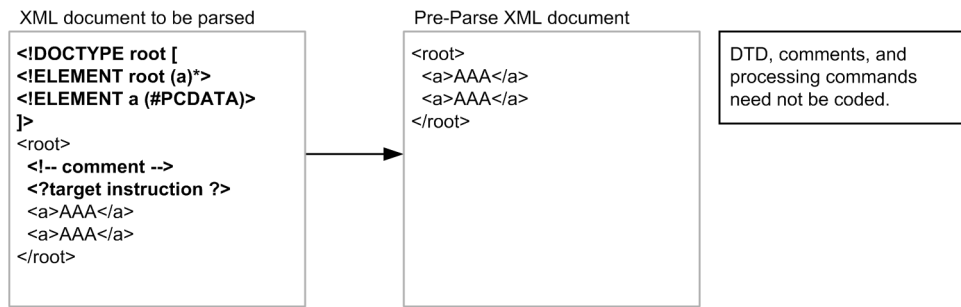
Figure 3–17: Example of coding of empty elements (policy for creating the Pre-Parse XML document)



(7) DTDs, comments, and processing instructions

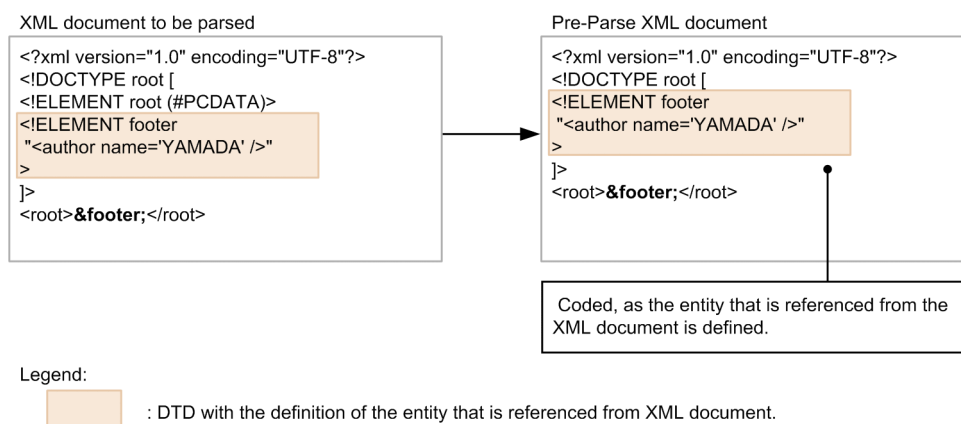
The DTDs, comments, and processing instructions are not included in the prepared object. Therefore, these might not be described in the Pre-Parse XML document (even if these are described, they will not have an impact on the prepared object that will be generated). The following figure shows an example:

Figure 3–18: Example related to DTDs, comments, and processing instructions (policy for creating the Pre-Parse XML document)



However, the DTDs that define the entities referenced from within the XML document for parsing must be described in the Pre-Parse XML document. Following figure shows an example:

Figure 3–19: Example of coding the definition of entities referenced from within the XML document (policy for creating the Pre-Parse XML document)



3.4.4 Generating the preparsed object

Parse the Pre-Parse XML document and generate the `PreparsedObject` instance that is the preparsed object.

To generate the preparsed object (`PreparsedObject` instance), use the following methods:

- `newInstance` method of the `com.cosminexus.jaxp.preparsedxml.PreparsedObjectFactory` class
- `newPreparsedObject` method of the `com.cosminexus.jaxp.preparsedxml.PreparsedObject` class

For details about the methods, see [3.4.7 Classes used in preparing the high-speed parse support function](#).

An example of code for generating the preparsed object is as follows:

```
// Prepare the Pre-Parse XML document
File xml = new File("learning1.xml");
// Prepare the entity resolver
MyEntityResolver entityResolver = new MyEntityResolver();
// Prepare the error handler
MyErrorHandler errorHandler = new MyErrorHandler();
```

```
// Generate PreparedObjectFactory
PreparedObjectFactory pof = PreparedObjectFactory.newInstance();
// Enable the namespace
pof.setNamespaceAware(true);
// Set up the entity resolver
pof.setEntityResolver(entityResolver);
// Set up the error handler
pof.setErrorHandler(errorHandler);
// Generate the prepared object
PreparedObject pobj = pof.newPreparedObject(xml);
```

3.4.5 Setting up the prepared object

Set up the prepared object in an XML parser of Cosminexus XML Processor. The prepared object can be set up in the following XML parsers:

- `javax.xml.parsers.DocumentBuilder`
- `javax.xml.parsers.SAXParser`
- `org.xml.sax.XMLReader`

To set up the prepared object in an XML parser, use the `http://cosminexus.com/xml/properties/preparedobject-load` property. For details about the property, see [4.6 How to use the properties of the high-speed parse support function](#).

3.4.6 Parsing the XML document

Parsing the XML document by the `parse` method of XML parser in which the prepared object is set up. The following table describes the `parse` methods for high-speed parse support function:



Important note

The operation is not guaranteed if the high-speed parse support function is used with a `parse` method that is not mentioned in the table.

Table 3–8: Parse methods with which the high-speed parse support function is used

Class name	Method name
<code>javax.xml.parsers.DocumentBuilder</code>	<code>parse(File f)</code>
	<code>parse(InputSource is)</code>
	<code>parse(InputStream is)</code>
	<code>parse(InputStream is, String systemId)</code>
	<code>parse(String uri)</code>
<code>javax.xml.parsers.SAXParser</code>	<code>parse(File f, DefaultHandler dh)</code>
	<code>parse(InputSource is, DefaultHandler dh)</code>
	<code>parse(InputStream is, DefaultHandler dh)</code>

Class name	Method name
	parse(InputStream is, DefaultHandler dh, String systemId)
	parse(String uri, DefaultHandler dh)
org.xml.sax.XMLReader	parse(InputSource input)
	parse(String systemId)

3.4.7 Classes used in preparing the high-speed parse support function

Use the following classes while preparing the high-speed parse support function:

- PreparedObjectFactory
- PreparedObject

Each class is as follows:

(1) PreparedObjectFactory class

(a) Description

Factory class used to generate the prepared object.

The PreparedObjectFactory class is not thread safe. Therefore, the same PreparedObjectFactory cannot be accessed simultaneously from more than one thread. To avoid conflicts between threads, use one of the following methods in PreparedObjectFactory class:

- Each thread has one PreparedObjectFactory instance.
- Each thread exclusively accesses the PreparedObjectFactory instance.

(b) Package and class names

com.cosminexus.jaxp.preparedxml.PreparedObjectFactory

(c) Format

```
public class PreparedObjectFactory
```

(d) List of methods

Method name	Function
<code>newInstance</code>	Generate and return the factory for generating the prepared object.
<code>setNamespaceAware</code>	Set up whether the namespace is enabled or disabled when the Pre-Parse XML document is parsed.
<code>setErrorHandler</code>	Set up the error handler for receiving the errors that occur when parsing the Pre-Parse XML document.
<code>setEntityResolver</code>	Set up the entity resolver for resolving the entities included in the Pre-Parse XML document.
<code>setTuningInfoFlag</code>	Set up whether to output the tuning information.
<code>newPreparedObject</code>	Parse the Pre-Parse XML document, and then generate the prepared object.

(e) Method details

newInstance method

Description

Generates and returns the factory for generating the prepared object.

Format

```
public static PreparedObjectFactory newInstance()
```

Parameter

None

Return value

This method returns the new instance of `PreparedObjectFactory`.

Exception

None

setNamespaceAware method

Description

Sets up whether the namespace is enabled or disabled when the Pre-Parse XML document is parsed.

By default, `false` is set up.

Format

```
public void setNamespaceAware(boolean awareness)
```

Parameter

- **awareness**

Sets up whether the namespace is enabled or disabled.

`true`: The namespace is enabled

`false`: The namespace is disabled

Return value

None

Exception

None

setErrorHandler method

Description

Sets up the error handler for receiving the errors that occur when parsing the Pre-Parse XML document when the `newInstance` method is invoked.

By default, the operation is performed considering that the following error handler is set up:

```
class DefaultErrorHandler implements ErrorHandler {
    public void fatalError(SAXParseException e) throws SAXException {
        throw e;
    }
    public void error(SAXParseException e) throws SAXException {
        throw e;
    }
    public void warning(SAXParseException e) throws SAXException {
        // No operations
    }
}
```

```
}  
}
```

Format

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

Parameter

- **errorHandler**

Sets up the error handler. When `null` is set up, the default operation is performed.

Return value

None

Exception

None

setEntityResolver method

Description

Sets up the entity resolver for resolving the entities included in the Pre-Parse XML document when the `newPreparsedObject` method is invoked.

By default, the operation is performed considering that the following entity resolver is set up:

```
class DefaultEntityResolver implements EntityResolver {  
    InputSource resolveEntity(String publicId, String systemId)  
    throws SAXException, IOException {  
        returns null; // Always returns null  
    }  
}
```

Format

```
public void setEntityResolver(org.xml.sax.EntityResolver entityResolve  
r)
```

Parameter

- **entityResolver**

Sets up the entity resolver. When `null`, the default operation is performed.

Return value

None

Exception

None

setTuningInfoFlag method

Description

Sets up whether to output the tuning information.

By default, `false` is set.

To output the tuning information, you must set `true` in this method, and `INFO` in the system property `com.cosminexus.jaxp.preparsedxml.tuning.level`. For details about how to output the tuning information, see [3.4.9\(3\) How to output the tuning information](#).

Format

```
public void setTuningInfoFlag(boolean state)
```

Parameter

- **state**
Sets up whether to output the tuning information.
`true`: The tuning information is output
`false`: The tuning information is not output

Return value

None

Exception

None

newPreparedObject method

Description

Parse the Pre-Parse XML document, and generates the prepared object.

Format

```
public PreparedObject newPreparedObject(java.io.File xml)
throws IllegalArgumentException, SAXException, IOException
```

Parameter

- **xml**
Specifies the `File` object that indicates the Pre-Parse XML document. You cannot specify `null`.

Return value

This method returns the new instance of `PreparedObject` that is the prepared object.

Exception

- **IllegalArgumentException**
This exception occurs when `null` is specified in the `xml` parameter.
- **SAXException**
This exception occurs in the following cases:
 - When an SAX error occurs while processing
 - When an attempt is made to pre-parse XML1.1 document (for details, see [6.19.3 Notes on XML1.1](#))
- **IOException**
This exception occurs when an I/O error occurs.

(2) PreparedObject class

(a) Description

This class indicates the prepared object.

(b) Package and class names

```
com.cosminexus.jaxp.preparedxml.PreparedObject
```

(c) Format

```
public class PreparedObject
```

(d) List of methods

Method name	Function
<code>isNamespaceAware</code>	Returns whether the namespace set up when parsing the Pre-Parse XML document is enabled or disabled.
<code>getErrorHandler</code>	Returns the error handler set up when parsing the Pre-Parse XML document.
<code>getEntityResolver</code>	Returns the entity resolver set up when parsing the Pre-Parse XML document.

(e) Method details

isNamespaceAware method

Description

Returns whether the namespace set up when parsing the Pre-Parse XML document is enabled or disabled.

Format

```
public boolean isNamespaceAware()
```

Parameter

None

Return value

`true`: The namespace is enabled

`false`: The namespace is disabled

Exception

None

getErrorHandler method

Description

Returns the error handler set up when parsing the Pre-Parse XML document.

Format

```
public ErrorHandler getErrorHandler()
```

Parameter

None

Return value

This method returns the error handler set up when parsing the Pre-Parse XML document. Returns `null` if this method is not set up.

Exception

None

getEntityResolver method

Description

Returns the entity resolver set up when parsing the Pre-Parse XML document.

Format

```
public EntityResolver getEntityResolver()
```

Parameter

None

Return value

This method returns the entity resolver set up when parsing the Pre-Parse XML document. Returns `null` if this method is not set up.

Exception

None

3.4.8 Coding example for using the high-speed parse support function

An example of code described in the user program for using the high-speed parse support function is as follows:

```
import java.io.File;
import java.io.IOException;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import org.xml.sax.EntityResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import com.cosminexus.jaxp.preparsedxml.PreparsedObjectFactory;
import com.cosminexus.jaxp.preparsedxml.PreparsedObject;

public class TestSAXParser{
    public static void main(String[] args){
        try{
            // Generate the preparsed object from the Pre-Parse XML document
            File xml = new File("learning1.xml");
            MyHandler handler = new MyHandler ();
            PreparsedObjectFactory pof = PreparsedObjectFactory.newInstance();
            pof.setNamespaceAware(true);
            pof.setEntityResolver(handler);
            pof.setErrorHandler(handler);
            PreparsedObject pobj = pof.newPreparsedObject(xml);

            // Generate the XML parser
            SAXParserFactory spf = SAXParserFactory.newInstance();
            spf.setNamespaceAware(true);
            SAXParser sp = spf.newSAXParser();

            // Set up the preparsed object in the XML parser, and parse
            sp.setProperty("http://cosminexus.com/xml/properties/preparsedobject-load",
                pobj);
            sp.parse("SampleSAX.xml", handler);
        } catch (IllegalArgumentException iae){
            System.out.println("MSG : " + iae.getMessage());
        } catch (SAXException se){
            System.out.println("MSG : " + se.getMessage());
        } catch (IOException ioe){
```

```

    System.out.println("MSG : " + ioe.getMessage());
} catch (Exception e) {
    e.printStackTrace();
}
}
}

class MyHandler extends DefaultHandler {
    //Describe the implementation of the error handler and entity resolver
}

```

3.4.9 Tuning the Pre-Parse XML document

If you reference the tuning information of the Pre-Parse XML document, you can examine whether the high-speed parse support function is running effectively. The parsing speed can be improved by optimizing the Pre-Parse XML document based on the examination results.

This subsection describes the types of the tuning information of Pre-Parse XML document, and also describes how to use and output the information.

(1) Types of tuning information

The following two types of files can be output as the tuning information:

- Tuning summary file
- Tuning details file

The information about the matching process when parsing is output to these files. The matching process is used to determine whether you can use the information about the preparsed object when parsing. The cases where you can use the information about the preparsed object are referred to as *successful in matching process*, while the cases where you cannot use the information about the preparsed object are referred to as *failed matching process*.

Each file is as follows:

(a) Tuning summary file

The success rate of the matching process, file names of the Pre-Parse XML document used to generate the preparsed object, system identifier of the XML document for parsing and tuning details file name is output for each parsing in which the high-speed parse support function is used. You can examine the extent of effective usage of the preparsed object in parsing with the high-speed parse support function.

For details about the tuning summary file, see [3.4.10 Details of the tuning summary file](#).

(b) Tuning details file

Besides the information about the tuning summary file, the history of the matching process is output for a single parsing in which the high-speed parse support function is used. For the parsing executed using the high-speed parse support function, you can examine the details of the location where the matching process has failed.

For details about the tuning details file, see [3.4.11 Details of the tuning details file](#).

(2) How to use the tuning information

The details on how to use the tuning information is as follows:

1. Output the tuning information.

For details about how to output the tuning information, see [3.4.9 \(3\) How to output the tuning information](#).

2. Reference the tuning summary file, and extract the parsing with a low success rate in matching process.

A low success rate of the matching process indicates that there is a vast difference between the structure of the Pre-Parse XML document, which is studied by the preparsed object, and the structure of the XML document for parsing, and the high-speed parse support function is not operating effectively.

3. Reference the tuning details file for each parsing extracted in step 2, and examine the history of the locations at which the matching process has failed.

Examine the history of the locations where the matching process has failed, and extract the parts where the structure of the Pre-Parse XML document does not match the structure of the XML document for parsing.

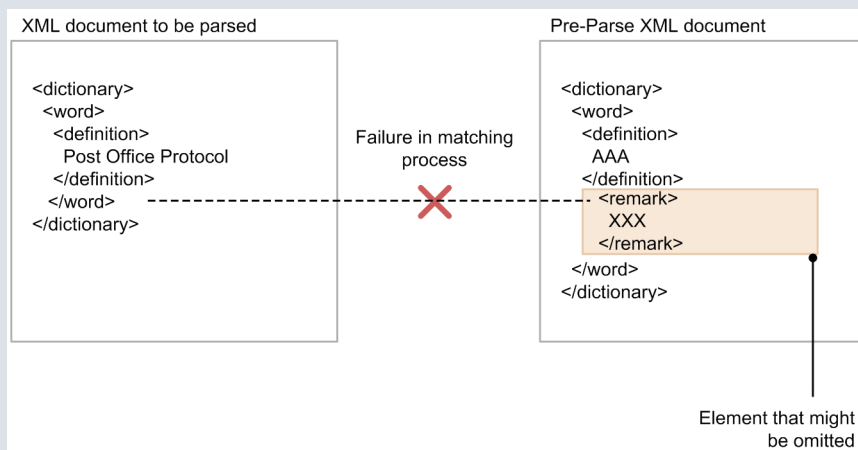
4. Revise the Pre-Parse XML document.

Revise the Pre-Parse XML document based on the examination results of step 3. For details about the policy for creating the Pre-Parse XML document, see [3.4.3 Creating the Pre-Parse XML document](#).

Important note

If the success rate of the matching process is low the effect of the high-speed parse support function will reduce, even if all elements and attributes described in the XML document for parsing are included in the Pre-Parse XML document.

An example when the success rate of the matching process is low even if all elements and attributes described in the XML document for parsing are included in the Pre-Parse XML document is as follows.



In this example, as the `remark` element is omitted in the XML document for parsing, the matching process for the `remark` element would fail, and the failure history will be recorded in the tuning details file. As a result, the success rate of the matching process will decline.

(3) How to output the tuning information

To output the tuning information, you must set up the system property provided by Cosminexus XML Processor. To output the tuning information with high-speed parse using `PreparsedObject`, besides the setup of the system property, you must set up `true` in the `setTuningInfoFlag` method of the `PreparsedObjectFactory` class,

and then generate `PreparedObject`. For details about the `PreparedObjectFactory` class, see [3.4.7\(1\) *PreparedObjectFactory* class](#).

The system property used to output the tuning information, the output destination of the tuning information, and the operation when an error occurs is as follows.

Important note

The parsing performance deteriorates when the tuning information is output. Therefore, output the tuning information only when evaluating the performance of the high-speed parse support function.

(a) System property used to output the tuning information

System property name

`com.cosminexus.jaxp.preparedxml.tuning.level`

Value

- When OFF or when this property is not set up: The tuning information is not output
- INFO: The tuning information is output

Assumed that OFF is specified when other than INFO in upper case is specified.

(b) Output destination of the tuning information

The tuning information is output when parsing with the high-speed parse support function terminates. The output destination of the tuning information is determined in the following order:

1. *Server-log-output-directory*[#]/jaxp

[#]: The *server-log-output-directory* differs depending on the server in use.

For the J2EE server

The directory specified in the `ejb.server.log.directory` key of the option definition file (`usrconf.cfg`) of J2EE server. By default, it is the following directory:

Cosminexus-work-directory/ejb/*server-name*/logs

For the Web container server

The directory specified in the `web.server.log.directory` key of the option definition file (`usrconf.cfg`) of Web container server. By default, it is the following directory:

Cosminexus-installation-directory/CC/web/containers/*server-name*/logs

2. *Directory-that-can-be-acquired-with-the-user.dir-system-property*/logs/jaxp

A file with the same name will be overwritten. If a file with the same name exists, save that file, as and when required. For details about the rules for file names, see [3.4.10\(1\) *File name \(tuning summary file\)*](#) and [3.4.11\(1\) *File name \(tuning details file\)*](#).

The tuning summary file and tuning details file are not deleted automatically. Delete these files as and when required. For details about the upper-limit value of the file size of the tuning summary file, see [3.4.10\(3\) *Upper-limit value of the file size*](#). For details about the upper-limit value of the total file size of tuning details file, see [3.4.11\(3\) *Upper-limit value of the total file size*](#).

(c) Operation when an error occurs

The operation performed when the tuning information cannot be output for occurrence of an error is as follows:

When an I/O error occurs

If an I/O error occurs during the output of the tuning summary file and tuning details file, these files will not be output. Also, the exception `IOException` occurs.

When an error occurs when parsing the XML document

The information will not be output to the tuning summary file. The information until the occurrence of the error will be output to the tuning details file.

3.4.10 Details of the tuning summary file

This subsection describes the file name, output format, and upper-limit of the file size of the tuning summary file.

(1) File name (tuning summary file)

```
csmjaxp_ppxml_summary.csv
```

(2) Output format (tuning summary file)

Output line	Output contents
First line	Header indicating the tuning summary file Cosminexus XML Processor Fast Parse Function Tuning Summary Information
Second line	Week, date and time on which the tuning summary file was created
Third line	Line header from the fourth line onwards matching rate,prepared XML,parsed XML,tuning detail file,date
Fourth line and thereafter	The following information is demarcated by a comma and output: 1. Success rate of the matching process $\text{Success frequency} \div \text{matching process execution frequency} \times 100$ 2. pre-parse XML file name [#] 3. System identifier of the XML document for parsing [#] If the system identifier is not set up, null will be output. 4. File name of the tuning details file [#] 5. Date and time of completion of parsing

[#]

Even if commas (,) are included in the output character string, they will be output as it is.

Output example

```
1  Cosminexus XML Processor  Fast Parse Function  Tuning Summary Information
2  Create Date: Tue Feb 06 20:57:35 JST 2007
3  matching rate,prepared XML,parsed XML,tuning detail file,date
4  45.5,D:\tr1.xml,file:/novelA.xml,csmjaxp_ppxml_tr1_novelistA_00.txt, Tue Feb 06 20:57:35 JST 2007
5  90.0,D:\tr1.xml,file:/novelB.xml,...
6  97.5,D:\tr2.xml,file:/comicA.xml,...
7  85.0,D:\tr2.xml,file:/comicB.xml,...
```

(3) Upper-limit value of the file size

1 MB[#]

[#]

If the file size exceeds 1MB, the data thereafter will not be added.

Note that you can change the upper-limit value of the file size with the following system property:

System property name

`com.cosminexus.jaxp.preparsedxml.tuning.summaryfile_maxsize`

Value

Specify an integer value of 0 or more. The unit is MB.

If this property is not set up, and if a value out of range is specified, 1 will be set up.

3.4.11 Details of the tuning details file

This subsection describes the file name, output format, and upper-limit of the file size of the tuning details file.

(1) File name (tuning details file)

```
csmjaxp_ppxml_{character string indicating the Pre-Parse XML document}_{character string indicating the Pre-Parse XML document}_{serial number}.txt
```

Character string indicating the Pre-Parse XML document

Indicates the character string obtained from the end part of the absolute path of Pre-Parse XML document and excluding the part after the last period (.).

Example: When the absolute path of the Pre-Parse XML document is `D:\home\xml\training\trl.xml`

The character string indicating the Pre-Parse XML document will be `trl`.

Character string indicating the XML document for parsing

Indicates the character string obtained from the end part of the system identifier of XML document for parsing and excluding the part after the last period (.).

Example: When the system ID of the XML document for parsing is `file:/home/xml/data/novelA.xml`

The character string indicating the XML document for parsing will be `novelA`.

Serial number

Serial number from 00 to 99. This number increases with each parsing.

Note that this number will be reset to 00 when the J2EE server or Web container is restarted. After 99 it would be reset to 00.

(2) Output format (tuning details file)

Output line	Output contents
First line	Header indicating the tuning details file Cosminexus XML Processor Fast Parse Function Tuning Detail Information
Second line	Date and time when the tuning details file was created
Third line	File name of the Pre-Parse XML document

Output line	Output contents
Fourth line	System identifier of the XML document for parsing If the system identifier is not set up, null will be output.
Fifth line	Header indicating the start of the matching information Matching Information start
Sixth line and thereafter	Matching information (history of the matching process) When the matching process is successful o:{element or attribute for which the matching process was executed}# When the matching process fails x: expected {pattern where matching was tried}# but {pattern that appears in the XML document for parsing}#
Second from the last line	Footer indicating the end of the matching information Matching Information end
Last line	Success rate of the matching process Success frequency ÷ matching process execution frequency × 100 (%) However, if the matching process is not executed even once, 0.0% will be output

#

Only a part of the element or attribute is output. The characters before and after the element or attribute are also output. For a specific example, see the output example:

Output example

1	Cosminexus XML Processor Fast Parse Function Tuning Detail Information	
2	Create Date: Wed Feb 07 13:12:42 JST 2007	
3	Prepared XML: d:\home\xml\training\training1.xml	
4	Parsed XML: word.xml	
5	Matching Information start	
6	o:note>	
7	o:</note>	
8	o:<word>●	Indicates success in matching process of element <word>.
9	o:<name is_acronym=	
10	o:>	
11	o:</name>	
12	o:<definition>	
13	o:</definition>	
14	o:<definition>	
15	o:</definition>	
16	x:expected <definition> but </word> ●	Indicates failure in matching process of element <definition>.
17	<w	
18	o:</word>	
19	o:<word>	
20	x:expected <name is_acronym= but <update date="200	
21	o:<update date=	
22	o:/>	
23	o:<name is_acronym=	
24	o:>	
25	o:</name>	
26	o:<definition>	
27	o:</definition>	
28	x:expected <definition> but </word> ●	The character after the element is output. In this case, the linefeed character <w after the </word> is output.
29	<w●	
30	o:</word>	
31	o:<word>	
32	x:expected <name is_acronym= but <update date="200	
33	o:<update date=	
34	x:expected /> but e	
35	o: editor=	
36	o:/>●	A part of the element is output. In this case, only the /> indicating the close tag is output.
37	o:<name>	
38	o:</name>	
39	o:<definition>	
40	o:</definition>	
41	x:expected <definition> but </word>	
42	</d	
43	o:</word>	
44	x:expected <word> but <abcdef ●	A part of the element is output. In this case, the characters ghi/> after <abcdef are not output.
45	x:expected not begin with a but a ●	Indicates failure in matching process as the character a appears.
46	Matching Information end	
47	Matching rate: 78.4%	

(3) Upper-limit value of the total file size

32 MB[#]

[#]

If the total file size of the output tuning details files exceeds 32 MB, no more tuning detail files will be generated thereafter.

For example, if the tuning detail files are generated in the following order of 1 to 5, and if the total file size of 1 to 4 files while the 4th file is being generated becomes 40 MB, which is greater than 32 MB, the fifth file will not be generated.

1. csmjasp_ppxml_trl_novelistA_00.txt (10MB)
2. csmjasp_ppxml_trl_novelistB_00.txt (10MB)
3. csmjasp_ppxml_trl_novelistC_00.txt (10MB)
4. csmjasp_ppxml_trl_novelistD_00.txt (10MB)

5. csmjasp_ppxml_trl_novelistE_00.txt (10MB)

Note that the upper-limit value of the total file size can be changed by the following system property.

System property name

`com.cosminexus.jasp.preparsedxml.tuning.detailfile_totalmaxsize`

Value

Specify an integer value as 0 or more. The unit is MB.

If this property is not set up, and if a value out of range is specified, 32 will be set up.

3.5 Apache-specific functions

In version 08-70 and later versions of Cosminexus XML Processor, you can set up Apache-specific features and properties corresponding to XML parsers and XSLT or XSLTC transformers. You can also use Apache-specific Namespace URIs.

This section introduces Apache-specific features, properties, and Namespace URIs that can be used. For details on the respective functions, see the Apache sites for Xerces2 Java and Xalan Java 2.

Important note

The functions described in this section are specific to Apache and are not covered by the JAXP standards. Therefore, the operations performed by using these functions are not guaranteed. To use these functions, you must adequately check the operations beforehand.

3.5.1 Features that can be set up

- The Apache-specific features that can be specified in XML parsers are as follows:
 - <http://apache.org/xml/features/allow-java-encodings>
 - <http://apache.org/xml/features/continue-after-fatal-error>
 - <http://apache.org/xml/features/disallow-doctype-decl>
 - <http://apache.org/xml/features/dom/create-entity-ref-nodes>
 - <http://apache.org/xml/features/dom/defer-node-expansion>
 - <http://apache.org/xml/features/dom/include-ignorable-whitespace>
 - <http://apache.org/xml/features/generate-synthetic-annotations>
 - <http://apache.org/xml/features/honour-all-schemaLocations>
 - <http://apache.org/xml/features/nonvalidating/load-dtd-grammar>
 - <http://apache.org/xml/features/nonvalidating/load-external-dtd>
 - <http://apache.org/xml/features/scanner/notify-builtin-refs>
 - <http://apache.org/xml/features/scanner/notify-char-refs>
 - <http://apache.org/xml/features/standard-uri-conformant>
 - <http://apache.org/xml/features/validate-annotations>
 - <http://apache.org/xml/features/validation/dynamic>
 - <http://apache.org/xml/features/validation/schema>
 - <http://apache.org/xml/features/validation/schema/augment-psvi>
 - <http://apache.org/xml/features/validation/schema/element-default>
 - <http://apache.org/xml/features/validation/schema/normalized-value>
 - <http://apache.org/xml/features/validation/schema/full-checking>
 - <http://apache.org/xml/features/validation/warn-on-duplicate-attdef>
 - <http://apache.org/xml/features/validation/warn-on-undeclared-edef>

- <http://apache.org/xml/features/warn-on-duplicate-entitydef>
- <http://apache.org/xml/features/xinclude>
- <http://apache.org/xml/features/xinclude/fixup-base-uris>
- <http://apache.org/xml/features/xinclude/fixup-language>
- The Apache-specific features that can be specified in XSLT or XSLTC transformers are as follows:
 - <http://xml.apache.org/xalan/features/incremental>
 - <http://xml.apache.org/xalan/features/optimize>

3.5.2 Properties that can be set up

- The Apache-specific properties that can be specified in XML parsers are as follows:
 - <http://apache.org/xml/properties/dom/current-element-node>
 - <http://apache.org/xml/properties/dom/document-class-name>
 - <http://apache.org/xml/properties/input-buffer-size>
 - <http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation>
 - <http://apache.org/xml/properties/schema/external-schemaLocation>
 - <http://apache.org/xml/properties/security-manager>
- The Apache-specific properties that can be specified in XSLT or XSLTC transformers are as follows:
 - <http://xml.apache.org/xalan/properties/source-location>

3.5.3 Namespace URIs that can be set up

- The Apache-specific Namespace URIs that can be used in XSLT or XSLTC transformers are as follows:
 - <http://xml.apache.org/xalan>
 - <http://xml.apache.org/xalan/java>
 - <http://xml.apache.org/xalan/PipeDocument>
 - <http://xml.apache.org/xalan/redirect>
 - <http://xml.apache.org/xalan/sql>
 - <http://xml.apache.org/xalan/xsltc>
 - <http://xml.apache.org/xalan/xsltc/java>
 - <http://xml.apache.org/xslt>
 - <http://xml.apache.org/xslt/java>

3.6 Correspondence Between Error Messages in XML Schema and W3C Specifications

Each error message output by Cosminexus XML Processor is provided with an identifier to identify the associated W3C specification. This is to provide consistent error information for the syntax error in the schema documents and the validation error in XML documents.

The list of error message identifiers is found in Part1: Structure Appendices C of the W3C XML Schema specification.

For example, if character-string data is specified in an XML document even though the integer data type (`xsd:integer`) is defined for an element in the schema document, the following message is output:

`KECX06036-E cvc-datatype-valid.1.2.1: 'ABC' is not a valid value for 'integer'.`

In this message, the identifier is `cvc-datatype-valid.1.2.1`. This indicates a violation of the rule described in *1.2.1 of Validation Rule:Datatype Valid*.

3.7 Parser Switching functionality

3.7.1 Overview of the Parser Switching functionality

The *Parser Switching functionality* switches the priority order of the XML implementation by XML Processor and the framework or user-specified XML implementation. The XML implementation can be switched to the one supplied with the framework by using this functionality.

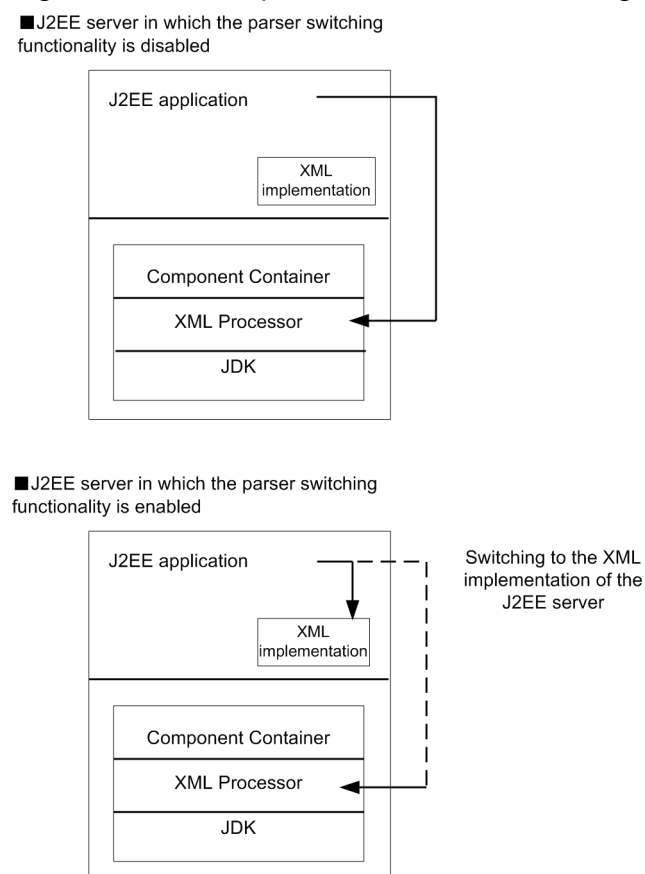
By using the Parser Switching functionality, the XML implementation can be switched even in the applications that use frameworks presuming other XML implementations. However, the XML implementation used by the Application Server components cannot be switched from XML Processor with the Parser Switching functionality.

(1) Scope of the Parser Switching functionality

The Parser Switching functionality is applicable to J2EE servers. The parser switching settings do not affect other J2EE servers.

Also, even if the Parser Switching functionality is enabled, XML Processor is used to start or deploy a J2EE server instead of the user-specified XML implementation. The following figure shows the scope of the Parser Switching functionality.

Figure 3–20: Scope of the Parser Switching functionality



(2) Types of the target XML implementations

The following table describes the methods used for switching the XML implementation in the JAXP, StAX, and JAXB APIs.

Table 3–9: Names of the methods for switching the XML implementation

No.	API type	Class name	Method name
1	JAXP API	<code>javax.xml.datatype.DatatypeFactory</code>	<code>newInstance()</code>
2		<code>javax.xml.parsers.DocumentBuilderFactory</code>	<code>newInstance()</code>
3		<code>javax.xml.parsers.SAXParserFactory</code>	<code>newInstance()</code>
4		<code>javax.xml.transform.TransformerFactory</code>	<code>newInstance()</code>
5		<code>javax.xml.transform.sax.SAXTransformerFactory</code>	<code>newInstance()</code>
6		<code>javax.xml.validation.SchemaFactory</code>	<code>newInstance(String schemaLanguage)</code>
7		<code>javax.xml.xpath.XPathFactory</code>	<code>newInstance()</code>
8			<code>newInstance(String URI)</code>
9		<code>org.w3c.dom.bootstrap.DOMImplementationRegistry</code>	<code>newInstance()</code>
10		<code>org.xml.sax.helpers.XMLReaderFactory</code>	<code>createXMLReader()</code>
11	StAX API	<code>javax.xml.stream.XMLInputFactory</code>	<code>newInstance()</code>
12			<code>newFactory()</code>
13		<code>javax.xml.stream.XMLOutputFactory</code>	<code>newInstance()</code>
14			<code>newFactory()</code>
15		<code>javax.xml.stream.XMLEventFactory</code>	<code>newInstance()</code>
16			<code>newFactory()</code>
17	JAXB API	<code>javax.xml.bind.JAXBContext</code>	<code>newInstance(Class... classesToBeBound)</code>
18			<code>newInstance(Class[] classesToBeBound, Map<String, ?> properties)</code>
19			<code>newInstance(String contextPath)</code>
20			<code>newInstance(String contextPath, ClassLoader classLoader)</code>
21			<code>newInstance(String contextPath, ClassLoader classLoader, Map<String, ?> properties)</code>

3.7.2 Usage of the Parser Switching functionality

The Parser Switching functionality includes the XML implementation used in an application and is set up in the following J2EE server definition files:

- User property file for J2EE servers (`usrconf.properties`)
- Security policy file for J2EE servers (`server.policy`)

For details on these files, see *2. Files Used in J2EE Servers* in the *uCosminexus Application Server Definition Reference Guide*.

The storage locations of these files are as follows:

- In Windows

```
Application-Server-installation-directory\CC\server\usrconf\ejb\server-name\
```

- In UNIX

```
/opt/Cosminexus/CC/server/usrconf/ejb/server-name/
```

(1) Including the XML implementation

To include the XML implementation, you store the XML implementation under the library JAR of a J2EE application, or WEB-INF/lib of a Web application that will switch the implementation. If you want to switch the XML implementation with multiple applications, include the XML implementation in the applications. For details on the library JAR, see *2.3.3 J2EE applications and J2EE component* in the *uCosminexus Application Server System Design Guide*.

Note that in the local call optimization functionality of Component Container, if `all` is specified in the optimization scope, see the notes described in *3.7.4(5) Notes on using the local call optimization functionality of Component Container*.

(2) Setting up the user property file for J2EE servers

The user property file for J2EE servers (`usrconf.properties`) defines the system properties to be specified for starting a J2EE server. To use the Parser Switching functionality, you must specify the properties described in the following table in the user property file for J2EE servers, according to the target XML implementation type.

Table 3–10: Properties to be specified in the user property file for J2EE servers

API type	Property name
JAXP API	<code>com.cosminexus.jaxp.change</code>
StAX API	<code>com.cosminexus.stax.change</code>
JAXB API	<code>com.cosminexus.jaxb.change</code>

The following table describes the values that can be specified in these properties and their meanings.

Table 3–11: Set values and meanings of the properties

Set value of the property	Meaning
<code>true</code>	The process switches to the XML implementation that includes the XML implementation for APIs.
<code>false</code> (Default value)	The XML implementation provided by XML Processor will be used.

(3) Setting up the security policy file for J2EE servers

To start a J2EE server without specifying the `-nosecurity` option (releasing `SecurityManager`), you specify the following settings in the security policy file for J2EE servers (`server.policy`) depending on the API type:

- If the switching of the JAXB API implementation is specified

```

permission java.lang.RuntimePermission "accessDeclaredMembers";
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
permission java.io.FilePermission "${cosminexus.home}/jaxp/lib/csmjaxb.jar
", "read";

```

3.7.3 Performance of the Parser Switching functionality

When you do not use the Parser Switching functionality, the processing has the same performance as in the case of version 09-00.

If the Parser Switching functionality is used, the processing speed reduces by a few milliseconds the first time when the `Factory` class instances described in [3.7.1\(2\) Types of the target XML implementations](#) are generated. From the second and subsequent times, the deterioration is not problematic. There is no performance deterioration for the `javax.xml.validation.SchemaFactory`, `javax.xml.xpath.XPathFactory`, and `javax.xml.bind.JAXBContext` APIs.

3.7.4 Notes on the Parser Switching functionality

(1) XML implementation for the container extension library

If you specify a class path for an XML implementation other than XML Processor in `usrconf.cfg`, the start processing of Application Server or operations of the components are affected, hence, you cannot specify the XML implementation in the container extension library.

For details on the container extension library, see *16. Container Extension Library* in the *uCosminexus Application Server Common Container Functionality Guide*.

(2) Notes on using the Application Server functionality

The functionality provided with Application Server includes functionality that requires XML Processor operations. Note that the functionality described in the following table might not operate normally if used concurrently with the Parser Switching functionality in the same application.

Table 3–12: Application Server functionality that does not support the switching of XML implementation

API type	Application Server functionality that does not support the switching of XML implementation
JAXP API	<ul style="list-style-type: none"> All the functions of the Web Sservice functionality (JAX-WS functionality, JAX-RS functionality, WS-RM1.1 functionality, and WS-RM1.2 functionality) All the functions for the communication between a Web Service and SOAP (existing functionality) All the functions of the JPA functionality All the functions of the JSF functionality All the functions of the JSTL functionality All the functions of the BeanValidation functionality All the functions of the CDI functionality All the functions of the security management (Web Service) functionality All the functionality of cFramework Reliable Messaging functionality for transmission between queues JAXB functionality of XML Processor

API type	Application Server functionality that does not support the switching of XML implementation
	<ul style="list-style-type: none"> • All the functionality of Service Coordinator
StAX API	<ul style="list-style-type: none"> • All the functions of the Web Service functionality (JAX-WS functionality, JAX-RS functionality, and WS-RM1.2 functionality) • All the functions of the JSTL functionality • All the functions of the BeanValidation functionality • JAXB functionality of XML Processor • All the functions of Service Coordinator
JAXB API	<ul style="list-style-type: none"> • All the functions of the Web Service functionality (JAX-WS functionality, JAX-RS functionality, and WS-RM1.2 functionality) • All the functions of the JSTL functionality • All the functions of the BeanValidation functionality • All the functions of the Service Coordinator

As a roundabout, consider dividing the components that require the user-specified XML implementation and the components that require the Application Server functionality into separate applications.

Also, on the J2EE servers that operate Service Coordinator, operate the applications that require the user-specified XML implementation on a J2EE server different than the user-specified XML implementation with Service Coordinator.

(3) Versions of XML implementation to be switched

Switching the XML implementation with the Parser Switching functionality does not switch APIs. Therefore, you cannot change the API version from the one supported by Application Server.

For example, if you switch the XML implementation conforming to a version different from one supported by Application Server, JAXP1.4/StAX1.0/JAXB2.2 supported by XML Processor 09-50 are used and the APIs added to the new version cannot be used for development and execution. Also, take precautions when you switch to an XML implementation of a different version because the operations might be affected by changes in the specifications and errors or exceptions might occur when the APIs are used.

(4) Switching the XML implementation class

JAXP1.4 and JAXB2.2 provide the mechanism for switching the implementation classes. The Parser Switching functionality cannot be used to override implementation class switchover that has already been made in any of the following ways:

- Specifying a factory class by using a system property
- Specifying a JAXP factory class in `jaxp.properties`
- Specifying a StAX factory class in `stax.properties`
- Specifying a JAXB factory class in `jaxb.properties`

For example, with JAXB1, which requires implementation classes to be specified in `jaxb.properties`, the Parser Switching functionality cannot be used to switch implementation classes.

(5) Notes on using the local call optimization functionality of Component Container

When an XML implementation is included with a J2EE application and if `all` is specified in the optimization scope for the local call optimization functionality of Component Container

(`ejbserver.rmi.localinvocation.scope=all`), the XML implementation of all the applications running on the same J2EE server switches. The EJB-JARs and library JARs of J2EE applications are loaded using the same class loader, so the classes with the same name are shared between the J2EE applications. If there are classes with the same names but different contents, according to the Java specifications, note that only the class loaded first is enabled.

For details on the local call optimization functionality, see *Appendix B.2 Class loader configuration for local call optimization* in the *uCosminexus Application Server Common Container Functionality Guide*.

(6) Switching the XML implementation of the client-operated Java applications

With Java applications, the XML implementation is switched by specifying the system properties in the user property file for Java applications, so the Parser Switching functionality is not supported.

For details on the user property file for Java applications, see *12.2.2 `usrconf.properties` (User property file for Java applications)* in the *uCosminexus Application Server Definition Reference Guide*.

3.8 Secure processing functionality

3.8.1 Overview

From version 1.3 onwards, JAXP provides secure processing functionality to achieve secure (safe) processing in XML parsing. This functionality suppresses processing of suspicious XML code that can trigger a denial-of-service (DoS) attack by placing restrictions on the entities defined in the DTD or the values of `maxOccurs` attributes specified in the XML schema. For the classes listed later, use the `setFeature` method to set a feature that enables the secure processing functionality. How to use the feature is described in the following table. For more details, see the JSR 206 Java™ API for XML Processing (JAXP) 1.4 specifications and the Javadoc documentation for the `setFeature` method of each listed class.

- `javax.xml.parsers.SAXParserFactory`
- `javax.xml.parsers.DocumentBuilderFactory`
- `javax.xml.transform.TransformerFactory`
- `javax.xml.validation.SchemaFactory`
- `javax.xml.xpath.XPathFactory`

Table 3–13: How to use the feature

Feature name	Operation	Value
<code>http://javax.xml.XMLConstants/feature/secure-processing</code>	Enable the functionality	<code>true</code>
	Disable the functionality	<code>false</code>

In XMLP 09-50-03 or earlier, the secure processing functionality was disabled by default. In XMLP 09-50-04 or later, however, the functionality is enabled by default for some API methods. For the items subject to limit value checking, the functionality has been enhanced so that the limit values can be changed, and more items can be subject to restrictions on entities. In addition, the system property for compatibility is provided to retain the system behavior shown before the functionality was supported.

3.8.2 Effective scope of the functionality

The following table shows whether the functionality is enabled or disabled by default for API methods that are executed.

Table 3–14: Whether the functionality is enabled or disabled by default

No.	Class name and interface name	Method name	Default	
			09-50-03 or earlier	09-50-04 or later
1	<code>javax.xml.parsers.SAXParser</code>	<code>parse</code>	Disabled	Enabled
2	<code>org.xml.sax.XMLReader</code>	<code>parse</code>	Disabled	Enabled
3	<code>org.xml.sax.XMLFilter</code>	<code>parse</code>	Disabled	Enabled [#]
4	<code>javax.xml.parsers.DocumentBuilder</code>	<code>parse</code>	Disabled	Enabled
5	<code>javax.xml.transform.TransformerFactory</code>	<code>newTemplates</code>	Disabled	Disabled

No.	Class name and interface name	Method name	Default	
			09-50-03 or earlier	09-50-04 or later
6	javax.xml.transform.TransformerFactory	getAssociatedStylesheet	Disabled	Disabled
7	javax.xml.transform.TransformerFactory	newTransformer	Disabled	Disabled
8	javax.xml.transform.sax.SAXTransformerFactory	newXMLFilter	Disabled	Disabled
9	javax.xml.transform.sax.SAXTransformerFactory	newTransformerHandler	Disabled	Disabled
10	javax.xml.transform.Transformer	transform	Disabled	Disabled
11	javax.xml.validation.SchemaFactory	newSchema	Disabled	Enabled
12	javax.xml.validation.Validator	validate	Disabled	Enabled
13	javax.xml.xpath.XPath	evaluate	Disabled	Disabled
14	javax.xml.xpath.XPathExpression	evaluate	Disabled	Disabled
15	com.cosminexus.jaxp.preparedxml.PreparedObjectFactory	newPreparedObject	Disabled	Enabled

#

The functionality is disabled for the parse method of XMLFilter created by newXMLFilter of the SAXTransformerFactory class.

3.8.3 Items that can be restricted

This section shows the items on which this functionality can place restrictions. The limit values can be changed for these items.

(1) Restrictions on entities

XMLP provides functions that place restrictions on expansion of entities. The following items can be restricted:

- Number of times an entity reference is expanded
- String length of a general entity
- String length of a parameter entity
- Total string length of expanded general entities and parameter entities

(a) Number of times an entity reference is expanded

The number of times an entity reference is expanded can be limited. If a violation of this restriction occurs during XML parsing, a KECX01225-E error is reported as an error notification (`fatalError` event) or exception.

The following shows an example of XML code subject to the restriction on the maximum number of times an entity reference is expanded.

```
<!DOCTYPE root [
  <!ENTITY x3 "abc">
  <!ENTITY x2 "&x3;&x3;">
  <!ENTITY x1 "&x2;&x2;">
```

```
]>  
<root>&x1;</root>
```

In the preceding example, the number of times an entity reference is expanded is seven (&x1; &x2; &x3; &x3; &x2; &x3; &x3;).

(b) String length of a general entity

The string length of a general entity can be limited. If a violation of this restriction occurs during XML parsing, a KECX01232-E error is reported as an error notification (`fatalError` event) or exception.

The following shows an example of XML code subject to the restriction on the maximum string length of a general entity.

```
<!DOCTYPE root [  
  <!ENTITY GE4 "hijk">  
  <!ENTITY GE3 "abcd">  
  <!ENTITY GE2 "efgh&GE3;ijk">  
  <!ENTITY GE1 "lmn">  
<root>&GE1; &GE2;</root>
```

In the preceding example, the functionality performs a length check on GE1 and GE2, which reference entities. For entity GE1, the length of the string `lmn` (3 characters) is checked. For entity GE2, the length of the string `efghabcdijk` (11 characters), which results from expansion of GE3, is checked. The length of GE3 itself is not checked. No length check is performed on GE4, for which no entity reference occurs.

(c) String length of a parameter entity

The length of a parameter entity after expansion can be limited. If a violation of this restriction occurs during XML parsing, a KECX01232-E error is reported as an error notification (`fatalError` event) or exception.

The following shows an example of XML code subject to the restriction on the maximum string length of a parameter entity.

```
<!ENTITY % PE2 "abcd">  
<!ENTITY % PE1 "%PE2;efgh">  
<!ENTITY GE1 "ij">  
<!ENTITY % PEG1 "&GE1;klmn">  
<!ENTITY % PEEEXT SYSTEM "ext.xml">
```

In the preceding example, the functionality performs length checks as follows: For parameter entity PE1, the length of the string `abcdefgh` (8 characters), which results from expansion of parameter entity PE2, is checked. For parameter entity PE2, the length of the string `abcd` (4 characters) is checked. For PEG1, which is a parameter entity that references general entity GE1, the length of the string `&GE1;klmn` (9 characters) is checked. This is because a general entity referenced by a parameter entity is not expanded. No length check is performed on PEEEXT, which is an external parameter entity that is not referenced.

(d) Total string length of expanded general entities and parameter entities

The total length of expanded general entities and parameter entities can be limited. If a violation of this restriction occurs during XML parsing, a KECX01233-E error is reported as an error notification (`fatalError` event) or exception.

The following shows an example of XML code subject to the restriction on the total length of expanded general entities and parameter entities.

```
<!ENTITY GE3 "abcd">
<!ENTITY GE2 "efgh&GE3;ijk">
<!ENTITY GE1 "lmn">
<!ENTITY % PE2 "abcd">
<!ENTITY % PE1 "%PE2;efgh">
```

In the preceding example, if only general entities GE1 and GE2 are referenced, the total length of the expansion results of GE1 and GE2 (referenced general entities) and PE1 and PE2 (defined parameter entities) is 26 as follows:

- GE1: lmn (number of characters: 3)
- GE2: efghabcdijk (number of characters: 11)
- PE1: abcdefgh (number of characters: 8)
- PE2: abcd (number of characters: 4)

(2) Restrictions on an XML schema content model

This functionality can place a restriction on the maximum number of nodes that are created in an XML schema content model. If a violation of this restriction occurs during XML parsing, a KECX06248-E error is reported as an error notification (fatalError event) or exception. The following shows an example of an XML schema subject to the restriction on the maximum number of nodes created in the XML schema content model.

```
<xsd:complexType>
  <xsd:sequence maxOccurs="2">
    <xsd:element name="child" maxOccurs="1500"/>
  </xsd:sequence>
</xsd:complexType>
```

As shown in the preceding example, if a large value is specified for maxOccurs in a complex-type global definition (outermost defined <xsd:complexType>) in an XML schema, an error occurs. Even if the value of maxOccurs is smaller than the limit value, the internal counter might exceed the limit value, depending on the conditions such as the nesting depth and the settings of other maxOccurs attributes specified elsewhere. If the limit value is exceeded in this case, an error occurs.

3.8.4 Using the functionality

(1) Changing limit values

You can use the following ways to change limit values of the secure processing functionality:

- Specifying system properties provided by Cosminexus XML Processor
- Specifying local properties provided by Cosminexus XML Processor

If both a system property and API-specific property are specified to change the value of the same item, the specification by the API-specific property prevails.

The following table lists the provided system properties.

Table 3–15: List of provided system properties

No.	System property name	Description	Specifiable value [#]	Default value
1	com.cosminexus.jaxp.secure_processing.entityExpansionLimit	Number of times an entity reference is expanded	0 to 2147483647	100000
2	com.cosminexus.jaxp.secure_processing.maxGeneralEntitySizeLimit	String length of a general entity	0 to 2147483647	0
3	com.cosminexus.jaxp.secure_processing.maxParameterEntitySizeLimit	String length of a parameter entity	0 to 2147483647	1000000
4	com.cosminexus.jaxp.secure_processing.totalEntitySizeLimit	Total string length of expanded entities	0 to 2147483647	50000000
5	com.cosminexus.jaxp.secure_processing.maxOccurLimit	Number of nodes in the content model	0 to 2147483647	3000

#

If 0 is specified, an error check based on the limit value is disabled.

If a value outside the range of specifiable values is specified, the default value is used.

The following table lists the provided local properties. The table also shows the values that can be specified for the local properties.

If an invalid value is specified, the value is not set. Therefore, the value that is set currently is not changed.

Table 3–16: List of provided local properties

No.	Property name	Description
1	http://cosminexus.com/xml/properties/entityExpansionLimit	Number of times an entity reference is expanded
2	http://cosminexus.com/xml/properties/maxGeneralEntitySizeLimit	String length of a general entity
3	http://cosminexus.com/xml/properties/maxParameterEntitySizeLimit	String length of a parameter entity
4	http://cosminexus.com/xml/properties/totalEntitySizeLimit	Total string length of expanded entities
5	http://cosminexus.com/xml/properties/maxOccurLimit	Number of nodes in the content model

The following table lists the methods that are used to set the local properties listed in the preceding table.

Before you set the local properties, make sure that the secure processing functionality is enabled.

Table 3–17: Methods for setting the local properties that set limit values

No.	Class name and interface name	Method
1	javax.xml.parsers.DocumentBuilderFactory	setAttribute
2	javax.xml.parsers.SAXParser	setProperty
3	org.xml.sax.XMLReader	setProperty
4	javax.xml.transform.TransformerFactory	setAttribute
5	javax.xml.validation.SchemaFactory	setProperty

In the `com.cosminexus.jaxp.preparedxml.PreparedObjectFactory` class, you cannot use local properties to set limit values. In this case, you must use system properties to set limit values.

(2) System property for compatibility

The system property for compatibility is provided so that the system behavior shown before this functionality was supported can be retained. The following table shows the specifications of the provided property. If an API-specific feature that enables this functionality is specified when this system property has been specified, the specification of the API-specific feature prevails.

Table 3–18: System property for compatibility

System property name	Description	Value [#]
com.cosminexus.jaxp.secure_processing.compatible	The default settings in version 09-50-04 or later are used (see Table 3-14).	false (default)
	The default settings in version 09-50-03 or earlier are used.	true

#

If a value other than the lowercase string `true` is specified, the system assumes that `false` is specified.

3.9 Compatibility option functionality

With XML Processor version 07-00 or later, some of the functionality and operations differ from the previous versions because of migration from JAXP1.2 to JAXP1.3. If you use the compatibility option functionality of XML Processor, you can perform the same operations as the previous versions.

3.9.1 Overview of the compatibility option functionality

With the *compatibility option functionality* of XML Processor, you can use the functionality for validated parsing compatibility that uses the XML schema. With the functionality for validated parsing compatibility, version 06-00 operations and version 07-00 or later operations are switched for the items described in the following table.

Table 3–19: Operations switched with the functionality for validated parsing compatibility

Items	XML Processor 06-00 operations	XML Processor 07-00 or later operations
When the schema document and instance document reference a URI with an invalid syntax	No exception is thrown.	The following exceptions are thrown: <ul style="list-style-type: none">• For operations with JDK 5 or with JDK 7 or earlier <code>IllegalArgumentException</code>• For operations with JDK6 <code>NullPointerException</code> or <code>IllegalArgumentException</code>
When a non-existing schema document is referenced	The KECX06069-E error (warning event) is not reported.	The KECX06069-E error (warning event) is reported.

3.9.2 How to set up the compatibility option functionality

(1) Files for setting up the compatibility option functionality

You can set up the compatibility option functionality in one of the following files according to the environment in use:

- User property file for J2EE servers

```
Application-Server-installation-directory/CC/server/usrconf/ejb/server-name/usrconf.properties
```

- User property file for Web container servers

```
Application-Server-installation-directory/CC/web/containers/server-name/usrconf/usrconf.properties
```

- User property file for EJB client applications

```
Directory-for-storing-user-defined-file/usrconf.properties
```

(2) Setting up the properties

Specify the properties in the file in the following format. The values of the compatibility option are only read once during startup. You cannot change the values after startup.

Property-name=Set-value

The following table describes the property names and the set values used in the functionality for validated parsing compatibility.

Table 3–20: Property names and set values used in the functionality for validated parsing compatibility

Items	Property name	Set value
When the schema and instance documents reference a URI with an invalid syntax	<code>com.cosminexus.jaxp.compat.schema_illegal_uri</code>	<ul style="list-style-type: none">• ON The same operations are executed as in version 06-00.• OFF, other characters The same operations are executed as in version 07-00 or later (default).
When a non-existing schema document is referenced	<code>com.cosminexus.jaxp.compat.schemalocation</code>	

4

How to Use Features and Properties

This chapter describes how to use the features and properties provided in JAXP and JAXB. This chapter describes how to use the features and properties defined in JAXP, and the features and properties related to the extended functions of Cosminexus XML Processor.

4.1 How to Use the SAX2 Features and Properties

The SAX2 standards define how to use the SAX parser's optional functions by using features and properties.

A feature or property name is a URI corresponding to an optional function of the SAX parser. The formats of the feature and property names are as follows:

Feature name = `http://xml.org/sax/features/ + feature ID`
Property name = `http://xml.org/sax/properties/ + Property ID`

Table 4-1 and Table 4-2 list the SAX2 features and SAX2 properties supported by Cosminexus XML Processor respectively.

Table 4–1: SAX2 features supported by Cosminexus XML Processor

No.	Feature name
1	<code>http://xml.org/sax/features/external-general-entitie</code>
2	<code>http://xml.org/sax/features/external-parameter-entities</code>
3	<code>http://xml.org/sax/features/namespaces</code>
4	<code>http://xml.org/sax/features/namespace-prefixes</code>
5	<code>http://xml.org/sax/features/use-attributes2</code>
6	<code>http://xml.org/sax/features/use-locator2</code>
7	<code>http://xml.org/sax/features/use-entity-resolver2</code>
8	<code>http://xml.org/sax/features/validation</code>

Table 4–2: SAX2 properties supported by Cosminexus XML Processor

No.	Property name
1	<code>http://xml.org/sax/properties/declaration-handler</code>
2	<code>http://xml.org/sax/properties/lexical-handler</code>

For the meaning of each feature and property, see the following parts in Javadoc of *Simple API for XML (SAX) 2.0.2 (sax2r3)*:

- *SAX2 Standard Feature Flags* in the `org.xml.sax` package
- *SAX2 Standard Handler and Property IDs* in the `org.xml.sax` package

4.1.1 How to use the SAX2 features

To set a SAX2 feature, use the `setFeature` method of the `org.xml.sax.xml.XMLReader` class. How to set a feature is shown below:

```
// Generate SAX parser
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();

// Acquire the XMLReader included in SAX parser
XMLReader reader = parser.getXMLReader();

// Set up the "validation" feature as enabled
reader.setFeature("http://xml.org/sax/features/validation", true);
```

4.1.2 How to use the SAX2 properties

To set a SAX2 property, use the `setProperty` method of the `org.xml.sax.xml.XMLReader` class. How to set a property is shown below.

```
// Generate the SAX parser
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();

// Acquire the XMLReader included in SAX parser
XMLReader reader = parser.getXMLReader();

// Register the DeclHandler during property settings.
DeclHandler handler = new MyDeclHandler();
reader.setProperty("http://xml.org/sax/properties/declaration-handler", handler);
```

4.2 How to use StAX properties

The StAX standards allow you to specify properties in `XMLInputFactory` and `XMLOutputFactory`. You can specify the following properties:

- `javax.xml.stream.isNamespaceAware`
- `javax.xml.stream.isCoalescing`
- `javax.xml.stream.isReplacingEntityReferences`
- `javax.xml.stream.isSupportingExternalEntities`
- `javax.xml.stream.reporter`
- `javax.xml.stream.resolver`
- `javax.xml.stream allocator`
- `javax.xml.stream.isRepairingNamespaces`

For meanings of the respective properties, see the following subsections in *Streaming API For XML Version 1.0 (JSR-173)*:

- *Chapter 4.5.1.1. Supported Properties of XMLInputFactory*
- *Chapter 4.5.2.1 Supported Properties of XMLOutputFactory*
- *Javadoc*

How to use StAX properties

To specify properties for `XMLInputFactory`, use the `setProperty` method of the `javax.xml.stream.XMLInputFactory` class, and to specify properties for `XMLOutputFactory`, use the `setProperty` method of the `javax.xml.stream.XMLOutputFactory` class.

The following points describe how to specify the respective properties:

- To specify properties for `XMLInputFactory`

```
XMLInputFactory xif = XMLInputFactory.newInstance();
// Enable the name space processing.
xif.setProperty("javax.xml.stream.isNamespaceAware", true);
```

- To specify properties for `XMLOutputFactory`

```
XMLOutputFactory xof = XMLOutputFactory.newInstance();
//Set the default prefix to be output.
xof.setProperty("javax.xml.stream.isRepairingNamespaces", true);
```


4.3 How to use XSLT features

The JAXP standard defines how to use the XSLT transformer's optional functions by using features.

A feature name is a URI corresponding to an optional function of the XSLT transformer. The following table lists the XSLT features defined in JAXP.

Table 4–3: XSLT features defined in JAXP

No.	Feature name
1	<code>http://javax.xml.transform.stream.StreamSource/feature</code>
2	<code>http://javax.xml.transform.stream.StreamResult/feature</code>
3	<code>http://javax.xml.transform.dom.DOMSource/feature</code>
4	<code>http://javax.xml.transform.dom.DOMResult/feature</code>
5	<code>http://javax.xml.transform.sax.SAXSource/feature</code>
6	<code>http://javax.xml.transform.sax.SAXResult/feature</code>
7	<code>http://javax.xml.transform.sax.SAXTransformerFactory/feature</code>
8	<code>http://javax.xml.transform.sax.SAXTransformerFactory/feature/xmlfilter</code>

For the meanings of the features, see the FEATURE field described in each class of Javadoc in the documentation *JSR 206 Java API for XML Processing(JAXP) 1.4*.

How to use the XSLT features

You cannot set any XSLT features. To get an XSLT feature, use the `getFeature` method of the `javax.xml.transform.TransformerFactory` class. The result of `getFeature` is always `true`.

4.4 How to Use the XML Schema Properties

By setting a property defined in JAXP, you can identify the schema language and document used in the schema validation. For details about properties, see *Properties For Enabling Schema Validation* in *Chapter 4. Plugability Layer* of the documentation *JSR 206 Java API for XML Processing (JAXP) 1.4*.

4.4.1 How to Set the XML Schema Properties for the DOM Parser

To set any XML Schema properties to the DOM parser, use the `setAttribute` method of the `javax.xml.parsers.DocumentBuilderFactory` class. How to set a property is shown below.

```
try {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setNamespaceAware(true);
    // 1. Specify the settings for validating consistency.
    dbf.setValidating(true);
    // 2. Set the schema language used for schema validation.
    dbf.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaLanguage",
        "http://www.w3.org/2001/XMLSchema");
    // 3. Specify the schema document.
    dbf.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaSource",
        "purchaseOrder.xsd");
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document doc = db.parse("purchaseOrder.xml");
} catch (Exception e) {
    e.printStackTrace();
}
```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Perform the setting that enables validation.

Set the argument of the `setValidating` method on the `javax.xml.parsers.DocumentBuilderFactory` class to `true`. This setting enables the XML document validation function.

2. Set the schema language to use for schema validation.

Using the `setAttribute` method of the `javax.xml.parsers.DocumentBuilderFactory` class, for the property string `"http://java.sun.com/xml/jaxp/properties/schemaLanguage"` set the value `"http://www.w3.org/2001/XMLSchema"`, which indicates that validation will follow the XML Schema specification.

3. Identify the schema document.

Using the `setAttribute` method of the `javax.xml.parsers.DocumentBuilderFactory` class, for the property string `"http://java.sun.com/xml/jaxp/properties/schemaSource"` specify the schema document used for validation. This specification is not necessary if the schema document is identified in the XML document.

4.4.2 How to Set the XML Schema Properties SAX Parser

To set any XML Schema properties to the SAX parser, use the `setProperty` method of the `javax.xml.parsers.SAXParser` class. How to set a property is shown below.

```

try {
    SAXParserFactory spf = SAXParserFactory.newInstance();
    spf.setNamespaceAware(true);
    //1. Specify the settings for validating consistency.
    spf.setValidating(true);
    SAXParser sp = spf.newSAXParser();
    // 2. Set the schema language used for schema validation.
    sp.setProperty("http://java.sun.com/xml/jaxp/properties/schemaLanguage",
        "http://www.w3.org/2001/XMLSchema");
    // 3. Specify the schema document.
    sp.setProperty("http://java.sun.com/xml/jaxp/properties/schemaSource",
        "purchaseOrder.xsd");
    XMLReader reader = sp.getXMLReader();
    reader.parse("purchaseOrder.xml");
} catch (Exception e) {
    e.printStackTrace();
}

```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Perform the setting that enables validation.

Set the argument of the `setValidating` method on the `javax.xml.parsers.SAXParserFactory` class to `true`. This setting enables the XML document validation function.

2. Set to comply with the XML Schema specification.

Using the `setProperty` method of the `javax.xml.parsers.SAXParser` class, for the property string `"http://java.sun.com/xml/jaxp/properties/schemaLanguage"` set the value `"http://www.w3.org/2001/XMLSchema"`, which indicates that validation will follow the XML Schema specification.

3. Identify the schema document.

Using the `setProperty` method of the `javax.xml.parsers.SAXParser` class, for the property string `"http://java.sun.com/xml/jaxp/properties/schemaSource"` specify the schema document used for validation. This specification is not necessary if the schema document is identified in the XML document.

4.4.3 How to Set the Schema Document in the XML Document

Each approach described in [4.4.1 How to Set the XML Schema Properties for the DOM Parser](#) and [4.4.2 How to Set the XML Schema Properties SAX Parser](#) sets the schema document by using the property. Alternatively, you can identify the schema document directly in the XML document to be validated. Note that if you identify the schema document both in the XML document and by using the property, the schema document identified by using the property is used.

To identify the schema document in the XML document to be validated, use the `xsi:schemaLocation` or `xsi:noNamespaceSchemaLocation` attribute. Which attribute to use depends on whether the namespace is defined in the schema document. The following table lists the types of schema document to identify and the attributes to use for them.

Table 4–4: Types of the schema document to identify and the attributes to use for them

Type of schema document to identify	Attribute to use
Schema document with the namespace defined	<code>xsi:schemaLocation</code>
Schema document with no namespace defined	<code>xsi:noNamespaceSchemaLocation</code>

When using the `xsi:schemaLocation` attribute, specify both the namespace URI of the elements and attributes declared in the schema document to identify and the name of the schema document to identify, separated by a space.

Both the `xsi:schemaLocation` and `xsi:noNamespaceSchemaLocation` attributes belong to the namespace `http://www.w3.org/2001/XMLSchema-instance`. Therefore, when these attributes are used to identify a schema document, the namespace `http://www.w3.org/2001/XMLSchema-instance` needs to be declared.

The follow shows an example of specifying a schema document by using the `xsi:schemaLocation`:

```
<? xml version="1.0"?>
<po:purchaseOrder xmlns:po=http://www.myshopping.com/schema/purchaseOrder
xmlns:psd="http://www.myshopping.com/schema/personalData"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ... 1.
xsi:schemaLocation="http://www.myshopping.com/schema/purchaseOrder
purchaseOrder.xsd" ... 2.
:
:
</po:purchaseOrder>
```

1. Declare the namespace of the `xsi:schemaLocation` attribute.

To use the `xsi:schemaLocation` attribute for specifying the schema document, declare this attribute's namespace `http://www.w3.org/2001/XMLSchema-instance`.

2. Identify the schema document.

Specify both the namespace URI of the elements and attributes declared in the schema document to identify and the name of the schema document, separated by a space.

The above example specifies `purchaseOrder.xsd` as the schema document and `http://www.myshopping.com/schema/purchaseOrder` as the namespace for elements and attributes declared in the schema document `purchaseOrder.xsd`.

4.5 How to use the properties of the Shift_JIS switch function

To use the Shift_JIS switch function, you need to set the following property defined independently by Cosminexus XML Processor.

Property name = `http://cosminexus.com/xml/properties/shift_jis_map`


Set a character encoding as the property value. The following table lists the property values to set and their meanings.

Table 4–5: Property values to set and their meanings

Value set in the property ^{#1}	Meaning
SJIS	Applies the SJIS encoding.
MS932	Applies the MS932 encoding.
Any other character string	Ignores the specification. ^{#2}

#1
The specified character string is not case-sensitive.

#2
If the specification is ignored, the last specified encoding is valid.

 **Important note**
If the Shift_JIS switch function is not specified, the default encoding is SJIS.

The Shift_JIS switch function property needs to be set for each instance of the classes and interfaces shown below. The following table lists the methods for setting the Shift_JIS switch function. This table lists the methods to use for setting the properties for each class and interface.

Table 4–6: Methods for setting the Shift_JIS switch function

Class and interface	Method
<code>javax.xml.parsers.DocumentBuilderFactory</code>	<code>setAttribute</code>
<code>javax.xml.parsers.SAXParser</code>	<code>setProperty</code>
<code>org.xml.sax.XMLReader</code>	<code>setProperty</code>
<code>javax.xml.parsers.TransformerFactory</code>	<code>setAttribute</code>

The following subsections describe the cases in which to use the Shift_JIS switch function for the DOM parser, SAX parser, and XSLT transformer.

4.5.1 How to Use the Shift_JIS Switch Function for the DOM Parser

This subsection describes the flow of basic coding that uses the Shift_JIS switch function in the DOM parser.

```
//1. Create DocumentBuilderFactory.
javax.xml.parsers.DocumentBuilderFactory factory =
    javax.xml.parsers.DocumentBuilderFactory.newInstance();
// 2. Switch Shift_JIS to MS932.
factory.setAttribute("http://cosminexus.com/xml/properties/shift_jis_map","MS932");
// 3. Create DocumentBuilder.
javax.xml.parsers.DocumentBuilder db = factory.newDocumentBuilder();
```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Create DocumentBuilderFactory.

Use the static method `newInstance()` of `javax.xml.parsers.DocumentBuilderFactory` to create the factory.

2. Switch Shift_JIS to MS932.

In the `setAttribute` method for the DOM parser setting, specify the first argument to the setting key `http://cosminexus.com/xml/properties/shift_jis_map` and the second argument to MS932.

3. Create DocumentBuilder.

Use the `newDocumentBuilder()` method to create `DocumentBuilder`.

4.5.2 How to Use the Shift_JIS Switch Function for the SAX Parser

This subsection describes the flow of basic coding that uses the Shift_JIS switch function in the SAX parser. The follow shows an example of the coding:

```
// 1. Create SAXParserFactory.
javax.xml.parsers.SAXParserFactory factory =
    javax.xml.parsers.SAXParserFactory.newInstance();
// 2. Create SAXParser.
javax.xml.parsers.SAXParser parser = factory.newSAXParser();
// 3. Create XMLReader.
org.xml.sax.XMLReader reader = parser.getXMLReader();
// 4. Switch Shift_JIS to MS932.
reader.setProperty("http://cosminexus.com/xml/properties/shift_jis_map","MS932");
```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Create SAXParserFactory.

Use the static method `newInstance()` of `javax.xml.parsers.SAXParserFactory` to create the factory.

2. Create SAXParser.

Use `newSAXParser()` to create the SAX parser.

3. Obtain the XMLReader.

Use `getXMLReader()` to obtain the XMLReader.

4. Switch Shift_JIS to MS932.

In the `setProperty` method for the SAX parser setting, specify the first argument to the setting key `http://cosminexus.com/xml/properties/shift_jis_map` and the second argument to MS932.

4.5.3 How to Use the Shift_JIS Switch Function for the XSLT Transformer

This subsection describes the flow of basic coding that uses the Shift_JIS switch function in the XSLT transformer. The follow shows an example of the coding:

```
// 1. Create TransformerFactory.
javax.xml.transform.TransformerFactory factory =
    javax.xml.transform.TransformerFactory.newInstance();
// 2. Switch Shift_JIS to MS932.
factory.setAttribute("http://cosminexus.com/xml/properties/shift_jis_map","MS932");
// 3. Create Transformer.
factory.newTransformer();
```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Create TransformerFactory.

Use the static method `newInstance()` of `javax.xml.transform.TransformerFactory` to create the factory.

2. Switch Shift_JIS to MS932.

In the `setAttribute` method for the XSLT transformer factory setting, specify the first argument to the setting key `http://cosminexus.com/xml/properties/shift_jis_map` and the second argument to MS932.

3. Create Transformer.

Use the `newTransformer()` method to create a transformer.

4.6 How to use the properties of the high-speed parse support function

To use the high-speed parse support function, you need to set the following property defined independently by Cosminexus XML Processor.

Property name = `http://cosminexus.com/xml/properties/preparedobject-load`

Set up either `null` or `PreparedObject` object in the property value. The following table lists the property values to set and their meanings.

Table 4–7: Value and meaning of the property to be set up (high-speed parse support function)

Value set in the property [#]	Meaning
<code>null</code>	The prepared object is not used when parsing.
<code>PreparedObject</code> object	The specified prepared object is used when parsing.

#

If no value is specified, it is assumed to be `null`.

If a value other than `null` and the `PreparedObject` object is set up, the following exceptions occur depending on the XML parser in which the property is set up:

- In `DocumentBuilderFactory`: `IllegalArgumentException`
- In `SAXParser` or `XMLReader`: `SAXNotSupportedException`

The properties of the high-speed parse support function must be set up in each object of the following XML parsers. The following table describes how to set up the prepared object. This table lists the methods to use for setting the properties for each class and interface.

Table 4–8: Methods for setting up the prepared object

Class and interface	Method
<code>javax.xml.parsers.DocumentBuilderFactory</code>	<code>setAttribute</code>
<code>javax.xml.parsers.SAXParser</code>	<code>setProperty</code>
<code>org.xml.sax.XMLReader</code>	<code>setProperty</code>

The details for using the high-speed parse support function in `DocumentBuilder`, `SAXParser`, and `XMLReader` are described in the following subsections with appropriate examples.

4.6.1 How to set up the prepared object in DocumentBuilder

A coding example to set up the prepared object in `DocumentBuilder` is as follows:


```
// 1. Generate preparsed objects.
File xml = new File(...);
PreparedObjectFactory pof = PreparedObjectFactory.newInstance();
pof.setNamespaceAware(...);
pof.setEntityResolver(...);
pof.setErrorHandler(...);
PreparedObject pobj = pof.newPreparedObject(xml);

// 2. Set preparsed objects in DocumentBuilder.
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setAttribute("http://cosminexus.com/xml/properties/preparsedobject-load", pobj);
DocumentBuilder db = dbf.newDocumentBuilder();
```

The following describes the coding example. Each item number corresponds to the number in the comments in the coding example.

1. Generate the preparsed object.

For details about how to generate the preparsed object, see [3.4.4 Generating the preparsed object](#).

2. Set up the preparsed object in DocumentBuilder.

In the first argument of the `setAttribute` method used to set up the preparsed object, specify the setup key `http://cosminexus.com/xml/properties/preparsedobject-load`, and in the second argument, specify `pobj` that indicates the `PreparedObject` object.

4.6.2 How to set up the preparsed object in SAXParser

A coding example to set up the preparsed object in `SAXParser` is as follows:

```
// 1. Generate preparsed objects.
File xml = new File(...);
PreparedObjectFactory pof = PreparedObjectFactory.newInstance();
pof.setNamespaceAware(...);
pof.setEntityResolver(...);
pof.setErrorHandler(...);
PreparedObject pobj = pof.newPreparedObject(xml);

// 2. Set preparsed objects in SAXParser.
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
sp.setProperty("http://cosminexus.com/xml/properties/preparsedobject-load", pobj);
```

1. Generate the preparsed object.

For details about how to generate the preparsed object, see [3.4.4 Generating the preparsed object](#).

2. Set up the preparsed object in SAXParser.

In the first argument of the `setProperty` method used to set up the preparsed object, specify the setup key `http://cosminexus.com/xml/properties/preparsedobject-load`, and in the second argument, specify `pobj` that indicates the `PreparedObject` object.

4.6.3 How to set up the preparsed object in XMLReader

A coding example to set up the preparsed object in `XMLReader` is as follows:

```
// 1. Generate preparsed objects.
File xml = new File(...);
PreparsedObjectFactory pof = PreparsedObjectFactory.newInstance();
pof.setNamespaceAware(...);
pof.setEntityResolver(...);
pof.setErrorHandler(...);
PreparsedObject pobj = pof.newPreparsedObject(xml);

// 2. Set preparsed objects in XMLReader.
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
XMLReader reader = sp.getXMLReader();
reader.setProperty("http://cosminexus.com/xml/properties/preparsedobject-load", pobj);
```

1. Generate the preparsed object.

For details about how to generate the preparsed object, see [3.4.4 Generating the preparsed object](#).

2. Set up the preparsed object in XMLReader.

In the first argument of the `setProperty` method used to set up the preparsed object, specify the setup key `http://cosminexus.com/xml/properties/preparsedobject-load`, and in the second argument, specify `pobj` that indicates the `PreparsedObject` object.

4.7 Using features and properties of the secure processing functionality

The secure processing functionality can be enabled by setting one of the items that are prescribed as features in JAXP. The following table lists features. For details about features, see *JSR 206 Java™ API for XML Processing (JAXP) 1.4*.

Table 4–9: List of features for the secure processing functionality

No.	Feature name
1	<code>http://javax.xml.XMLConstants/feature/secure-processing</code>

The following table lists the properties that specify the limit values for the secure processing functionality in Cosminexus XML Processor.

For details about the functionality, see [3.8 Secure processing functionality](#).

Table 4–10: List of properties for the secure processing functionality

No.	Property name
1	<code>http://cosminexus.com/xml/properties/entityExpansionLimit</code>
2	<code>http://cosminexus.com/xml/properties/maxGeneralEntitySizeLimit</code>
3	<code>http://cosminexus.com/xml/properties/maxParameterEntitySizeLimit</code>
4	<code>http://cosminexus.com/xml/properties/totalEntitySizeLimit</code>
5	<code>http://cosminexus.com/xml/properties/maxOccurLimit</code>

4.8 How to use the properties of JAXB

In JAXB, you can set up the properties in Marshaller. The JAXB properties supported by Cosminexus XML Processor are as follows:

JAXB properties supported by Cosminexus XML Processor

- `jaxb.encoding`
- `jaxb.formatted.output`
- `jaxb.schemaLocation`
- `jaxb.noNamespaceSchemaLocation`
- `jaxb.fragment`

For details about properties, see *Chapter 4.5.2. Marshalling Properties* and *javadoc* of *JSR 222 The Java Architecture for XML Binding(JAXB) 2.2*.

For details about the character encodings that can be specified in `jaxb.encoding`, see *1.3.2 Character encodings that can be processed*.

To set up properties in Marshaller, use the `setProperty` method of the `javax.xml.bind.Marshaller` class. How to set a property is shown below.

```
// Generate the content tree for marshall.
JAXBContext jc = JAXBContext.newInstance("com.acme.foo");
Unmarshaller u = jc.createUnmarshaller();
Object topelement = u.unmarshal( new File( "foo.xml" ) );
// Generate Marshaller.
Marshaller m = jc.createMarshaller();
// Specify Shift_JIS to output encoding.
m.setProperty(Marshaller.JAXB_ENCODING, "Shift_JIS");
// Format the XML data using linefeed and indentation.
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
// Specify the xsi:schemaLocation property of XML data to be output.
m.setProperty(Marshaller.JAXB_SCHEMA_LOCATION, "http://foo/ns fooSchema.xsd");
// Marshall the content tree.
m.marshal(topelement, new FileOutputStream("outputFile.xml"));
```

5

How to Create a Program

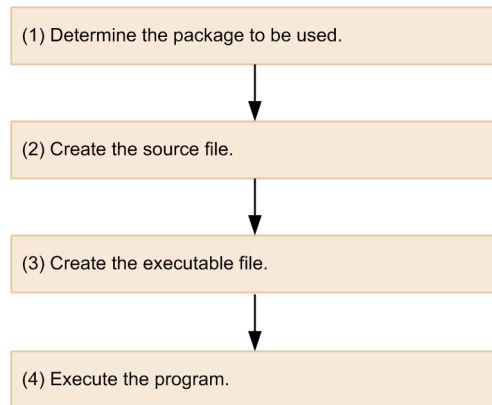
This chapter describes the general procedure for creating a program, packages to be used, and sample programs that use the key JAXP functions.

5.1 General Procedure for Creating a Program

5.1.1 Procedure for creating a program using Cosminexus XML Processor

The following figure shows the general procedure for creating a program by using Cosminexus XML Processor:

Figure 5–1: Procedure for creating a program using Cosminexus XML Processor



(1) Deciding which package to use

Cosminexus XML Processor provides the packages defined in the JAXP specification, the packages of XSLTC transformer function, and the packages defined in the JAXB specification. When creating a program, select a suitable package for use with the program.

For instance, use DOM when it is necessary to maintain the XML document in memory to change its structure or perform complex operations; use SAX when the processing is performed while interactively parsing the XML document, and use XSLT when the XML document is transformed into any other format such as HTML.

For details about packages provided in the JAXP specifications, see [2.2 JAXP-defined Packages and Their Functionality](#), for packages of the XSLTC transformer function, see [3.2.3 Class Used by the XSLTC Transformer](#), and for packages stipulated in the JAXB specifications, see [2.4 JAXB-defined Packages and Their Functionality](#).

(2) Creating source files

After deciding the package to use, create source files using the classes and interfaces contained in each package.

(3) Creating an executable file

Create an executable file from the source files you created.

To create an executable file, you need to compile the created source files by the `javac` command and link any libraries necessary for the classes and interfaces to be used.

(4) Executing the program

Execute the program you have compiled.

5.2 Package Names Used by Cosminexus XML Processor

Cosminexus XML Processor uses package names starting with the following strings:

- `com.cosminexus.jaxp.`
- `com.cosminexus.jaxb.`
- `com.cosminexus.stax.`
- `com.sun.istack.`
- `com.sun.xml.bind.`
- `com.sun.xml.bind.api.`
- `com.sun.xml.bind.marshaller.`
- `com.sun.xml.bind.v2.model.annotation.`
- `com.sun.xml.bind.v2.model.core.`
- `com.sun.xml.bind.v2.model.runtime.`
- `org.apache.xerces.util.`



Important note

Make sure that the package names of user programs do not begin with any of the preceding strings.

5.3 Troubleshooting Program Execution

When operating a system that uses Cosminexus XML Processor, if an error occurs that the system administrator needs to be notified of, or the appropriate countermeasure for an error is unclear, the following information used to investigate the failure must be gathered from application logs.

- Date and time when the failure occurred
- Name of the input XML documents when the failure occurred (including the schema document name and the stylesheet name)

Obtain the following failure information together with the application log above, and then contact the system administrator.

- Application log files
- The input XML documents when the failure occurred (including the schema document and the stylesheet)
- Cosminexus XML Processor error messages output when the failure occurred
- System properties and system class paths set on the server and client
- Server and client standard output and standard error output
- Cosminexus Component Container log and Web server log

- The information, as defined for Cosminexus Component Container, to be obtained when a failure occurs

The following shows the information, as defined for Cosminexus Component Container, to be obtained when a failure occurs.

- User definition file of the J2EE server
- Maintenance information of the J2EE server
- User definition file of the server management commands
- Maintenance information of server management commands
- Standard output and standard error output of the J2EE server
- Cosminexus TPBroker log on the J2EE server
- System properties and system class paths set in EJB client application
- Standard output and standard error output of the EJB client application
- Packet content collected by the network monitor tool

For details on how to obtain the failure information provided in Cosminexus Component Container, see *2.3 Acquiring the Data* in the *uCosminexus Application Server Maintenance and Migration Guide*.

5.4 Sample Program that Uses the DOM Parser

This section describes a sample program that uses the DOM parser.

The sample program is provided only in Windows.

The installation folder and file name of the sample program are as follows.

Installation folder:

installation-folder-of-Cosminexus-XML-Processor\samples\DOM

File name:

SampleDOM.java

5.4.1 Processing Performed by the Sample Program

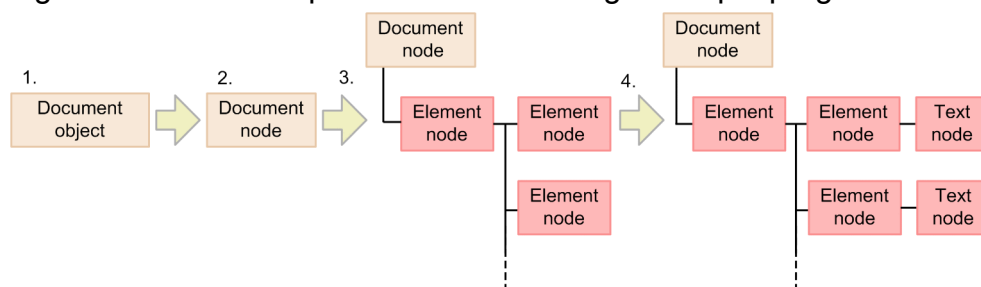
The sample program performs the following processing:

1. The program creates a new DOM tree that codes the product information.
2. The program transforms the created DOM tree into an XML document and outputs the XML document.
3. The program reads the output XML document.
4. From the read XML document, the program outputs the total cost of products to the window.

5.4.2 General Procedure for Creating the Sample Program

The following figure shows the general procedure for creating a sample program that uses the DOM parser.

Figure 5–2: General procedure for creating a sample program that uses the DOM parser



1. Create a Document object.
Create a new Document object.
2. Create the Document node.
By using the Document object you created in step 1, create the Document node that is the parent of the DOM tree.
3. Create the Element nodes.
Create the necessary Element nodes from the Document node to create the DOM tree.
4. Create the Text nodes.

Create the character strings that are the contents of the elements.

5.4.3 Sample program (SampleDOM.java)

The following shows the sample program:

```
import java.io.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import org.xml.sax.SAXException;
import org.w3c.dom.*;

public class SampleDOM
{
    public static final void main(String[] args)
    {
        try{
            new SampleDOM().CreateXMLFile();
            new SampleDOM().GetStringFromXML();
        }catch(Exception exception){
            exception.printStackTrace();
        }
    }

    //create XML file
    public final void CreateXMLFile()
    {
        try{
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();

            Document doc = db.newDocument();
            Element parent = doc.createElement("Payment_Slip");

            Element sub = doc.createElement("Merchandise");
            Element leaf1 = doc.createElement("Brand_Name");
            Element leaf2 = doc.createElement("Unit_Price");
            Element leaf3 = doc.createElement("Amount");
            leaf1.appendChild(doc.createTextNode("Television"));
            sub.appendChild(leaf1);
            leaf2.appendChild(doc.createTextNode("15000"));
            sub.appendChild(leaf2);
            leaf3.appendChild(doc.createTextNode("14"));
            sub.appendChild(leaf3);
            parent.appendChild(sub);

            sub = doc.createElement("Merchandise");
            leaf1 = doc.createElement("Brand_Name");
            leaf2 = doc.createElement("Unit_Price");
            leaf3 = doc.createElement("Amount");
            leaf1.appendChild(doc.createTextNode("Radio"));
            sub.appendChild(leaf1);
```

```

leaf2.appendChild(doc.createTextNode("3500"));
sub.appendChild(leaf2);
leaf3.appendChild(doc.createTextNode("6"));
sub.appendChild(leaf3);
parent.appendChild(sub);

sub = doc.createElement("Merchandise");
leaf1 = doc.createElement("Brand_Name");
leaf2 = doc.createElement("Unit_Price");
leaf3 = doc.createElement("Amount");
leaf1.appendChild(doc.createTextNode("Video"));
sub.appendChild(leaf1);
leaf2.appendChild(doc.createTextNode("21000"));
sub.appendChild(leaf2);
leaf3.appendChild(doc.createTextNode("4"));
sub.appendChild(leaf3);
parent.appendChild(sub);

doc.appendChild(parent);

OutputStream os =
    new BufferedOutputStream(
        new FileOutputStream("SampleDOM.xml"));
TransformerFactory tf =
    TransformerFactory.newInstance();
Transformer tran = tf.newTransformer();
tran.setOutputProperty("encoding", "Shift_JIS");
tran.setOutputProperty("indent", "yes");
tran.transform(new DOMSource(doc),
    new StreamResult(os));

os.flush();
os.close();

}catch(ParserConfigurationException e){
    e.printStackTrace();
}catch(TransformerConfigurationException e){
    e.printStackTrace();
}catch(TransformerException e){
    e.printStackTrace();
}catch(IOException e){
    e.printStackTrace();
}
}

//acquire data from the XML file made by CreateXMLFile, and calculate
public final void GetStringFromXML()
{
    try{
        int tvPrice = 0;
        int radioPrice = 0;
        int videoPrice = 0;

        DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();

        Document doc = builder.parse("SampleDOM.xml");

```

```

Element element1 = doc.getDocumentElement();
NodeList nodelist1 = element1.getElementsByTagName("Merchandise");

//loop for number of the merchandise
for(int nShouhin = 0 ;
    nShouhin < nodelist1.getLength() ; nShouhin++){
    String str = new java.lang.String();
    int price = 0;
    int number = 0;
    Element element2 = (Element)nodelist1.item(nShouhin);

    //get the brand name
    NodeList nodelist2 =
        element2.getElementsByTagName("Brand_Name");
    Node node1 = nodelist2.item(0);
    if(node1.getNodeType() == Node.ELEMENT_NODE){
        Element element3 = (Element)node1;
        NodeList nodelist3 = element3.getChildNodes();
        Node node2 = nodelist3.item(0);
        str = node2.getNodeValue();
    }

    //get the unit price of a brand name
    nodelist2 = element2.getElementsByTagName("Unit_Price");
    node1 = nodelist2.item(0);
    if(node1.getNodeType() == Node.ELEMENT_NODE){
        Element element3 = (Element)node1;
        NodeList nodelist3 = element3.getChildNodes();
        Node node2 = nodelist3.item(0);
        price = Integer.parseInt(node2.getNodeValue());
    }

    //get the amount of a brand name
    nodelist2 = element2.getElementsByTagName("Amount");
    node1 = nodelist2.item(0);
    if(node1.getNodeType() == Node.ELEMENT_NODE){
        Element element3 = (Element)node1;
        NodeList nodelist3 = element3.getChildNodes();
        Node node2 = nodelist3.item(0);
        number = Integer.parseInt(node2.getNodeValue());
    }

    //subtotal calculation of each merchandise
    if(str.equals("Television")){
        tvPrice = price * number;
    }else if(str.equals("Radio")){
        radioPrice = price * number;
    }else if(str.equals("Video")){
        videoPrice = price * number;
    }
}
int sum = tvPrice + radioPrice + videoPrice;
System.out.println(sum + " yen in total.");
}catch(ParserConfigurationException e){
    e.printStackTrace();
}catch(SAXException e){
    e.printStackTrace();
}catch(IOException e){

```

```
        e.printStackTrace();
    }
}
```

5.4.4 Execution Result of the Sample Program (SampleDOM.java)

The execution result of this sample program is output to the standard output and SampleDOM.xml.

Content of the standard output:

```
$ The total is 315000 yen.
```

Content output to SampleDOM.xml:

```
<?xml version="1.0" encoding="Shift_JIS"?>
<Payment_Slip>
<Merchandise>
<Brand_Name>Television</Brand_Name>
<Unit_Price>15000</Unit_Price>
<Amount>14</Amount>
</Merchandise>
<Merchandise>
<Brand_Name>Radio</Brand_Name>
<Unit_Price>3500</Unit_Price>
<Amount>6</Amount>
</Merchandise>
<Merchandise>
<Brand_Name>Video</Brand_Name>
<Unit_Price>21000</Unit_Price>
<Amount>4</Amount>
</Merchandise>
</Payment_Slip>
```

5.5 Sample Program that Uses the SAX Parser

This section describes a sample program that uses the SAX parser.

The sample program is provided only in Windows.

The installation folder and file name of the sample program are as follows.

Installation folder:

installation-folder-of-Cosminexus-XML-Processor\samples\SAX

File name:

- SampleSAX.xml
- SampleSAX.java

5.5.1 Processing Performed by the Sample Program

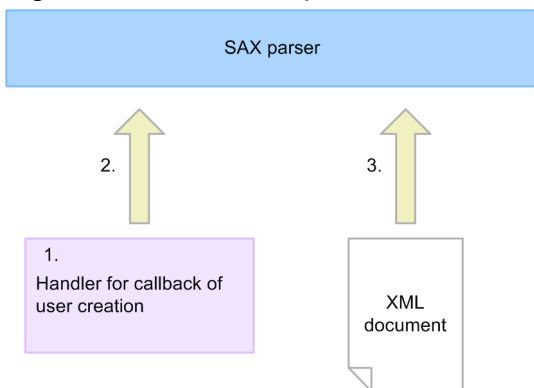
The sample program performs the following processing:

1. The program reads the XML document that codes the product information.
2. From the read XML document, the program outputs the total cost of products to the window.

5.5.2 General Procedure for Creating the Sample Program

The following figure shows the general procedure for creating a sample program that uses the SAX parser.

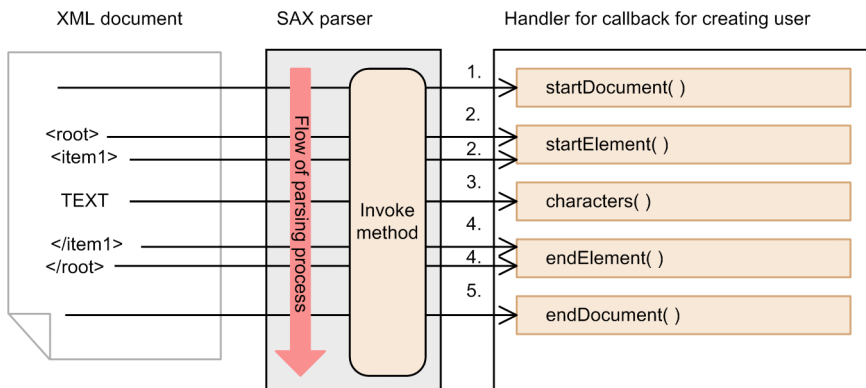
Figure 5–3: General procedure for creating a sample program that uses the SAX parser



1. Create the callback handler.
When parsing is started using the SAX parser, the information in the XML document is reported from the SAX parser. Create the callback handler to receive the information.
2. Register the callback handler with the SAX parser.
Register the callback handler you created in step 1 with the SAX parser.
3. Pass the XML document to the SAX parser to parse it.
Parse the XML document.

Methods called by the SAX parser:

The following diagram shows the methods called by the SAX parser:



1.startDocument method

The startDocument method is invoked when starting the parsing of XML document.
Code the default processing required in this method.

2.startElement method

The startElement method is invoked when searching for the open tag of element when parsing.

3.characters method

When a text enclosed in tag exists in the XML document, the characters method is invoked and string is acquired.

4.endElement method

The endElement method is invoked when searching for the close tag during parsing.

5.endDocument method

The endDocument method is invoked when completing parsing.

5.5.3 XML Document to Use (SampleSAX.xml)

The following shows the XML document to use:

```
<?xml version="1.0" encoding="Shift_JIS"?>
<Payment_Slip>
  <Merchandise>
    <Brand_Name>Television</Brand_Name>
    <Unit_Price>15000</Unit_Price>
    <Amount>14</Amount>
  </Merchandise>
  <Merchandise>
    <Brand_Name>Radio</Brand_Name>
    <Unit_Price>3500</Unit_Price>
    <Amount>6</Amount>
  </Merchandise>
  <Merchandise>
    <Brand_Name>Video</Brand_Name>
    <Unit_Price>21000</Unit_Price>
    <Amount>4</Amount>
  </Merchandise>
</Payment_Slip>
```

5.5.4 Sample Program (SampleSAX.java)

The following shows the sample program:

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.xml.sax.SAXException;
import java.io.*;

public class SampleSAX{

    public static final void main(String[] args){
        try{
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser parser = factory.newSAXParser();
            String xml = "SampleSAX.xml";
            DefaultHandler handler = new EventHandler();

            parser.parse(xml, handler);
        }catch(ParserConfigurationException e){
            e.printStackTrace();
        }catch(IOException e){
            e.printStackTrace();
        }catch(SAXException e){
            e.printStackTrace();
        }
    }
}

//calculate an amount of money in total and output it
class EventHandler extends DefaultHandler{

    int      tvPrice;           //Television unit price
    int      radioPrice;        //Radio unit price
    int      videoPrice;        //Video unit price
    int      tvNum;             //Television amount
    int      radioNum;          //Radio amount
    int      videoNum;          //Video amount
    boolean  tvFlag;            //Television flag
    boolean  radioFlag;         //Radio flag
    boolean  videoFlag;         //Video flag
    boolean  tankaFlag;         //Unit price flag
    boolean  volFlag;           //Amount flag
    boolean  nameFlag;          //Brand Name flag

    String str;                 //Character string of element contents
    StringBuffer buffer;        //buffer for character string connections of element contents

    //start notice event of XML
    public void startDocument(){
        tvPrice = 0;
        radioPrice = 0;
        videoPrice = 0;
        tvNum = 0;
        radioNum = 0;
    }
}
```



```

    videoNum = 0;
    tvFlag = false;
    radioFlag = false;
    videoFlag = false;
    tankaFlag = false;
    volFlag = false;
    nameFlag = false;
}

//end notice event of XML
public void endDocument(){
    int sum = tvNum * tvPrice + radioNum * radioPrice +
        videoNum * videoPrice;
    System.out.println(sum + " yen in total.");
}

//element start (a start of a tag) notice event
public void startElement(String nameSpace, String localName,
    String modName, Attributes attr){
    buffer = new StringBuffer();
    if(modName.equals("Unit_Price")){
        tankaFlag = true;
    }
    if(modName.equals("Amount")){
        volFlag = true;
    }
    if(modName.equals("Brand_Name")){
        nameFlag = true;
    }
}

//element end (the end of a tag) notice event
public void endElement(String nameSpace, String localName,
    String modName){
    str = buffer.toString();
    if(nameFlag == true){
        if(str.equals("Television")){
            tvFlag = true;
        }
        if(str.equals("Radio")){
            radioFlag = true;
        }
        if(str.equals("Video")){
            videoFlag = true;
        }
    }else if(tvFlag == true){
        if(tankaFlag == true){
            //set unit price of television
            tvPrice = Integer.parseInt(str);
        }else if(volFlag == true){
            //set amount of television
            tvNum = Integer.parseInt(str);
        }
    }else if(radioFlag == true){
        if(tankaFlag == true){
            //set unit price of radio
            radioPrice = Integer.parseInt(str);
        }else if(volFlag == true){

```

```

        //set amount of radio
        radioNum = Integer.parseInt(str);
    }
} else if (videoFlag == true) {
    if (tankaFlag == true) {
        //set unit price of video
        videoPrice = Integer.parseInt(str);
    } else if (volFlag == true) {
        //set amount of video
        videoNum = Integer.parseInt(str);
    }
}
}
if (modName.equals("Unit_Price")) {
    tankaFlag = false;
}
if (modName.equals("Amount")) {
    volFlag = false;
}
if (modName.equals("Brand_Name")) {
    nameFlag = false;
}
if (modName.equals("Merchandise")) {
    tvFlag = false;
    radioFlag = false;
    videoFlag = false;
}
}

//notice of character string event
public void characters(char[] ch, int start, int length) {
    buffer.append(ch, start, length);
}
}

```

5.5.5 Execution Result of the Sample Program (SampleSAX.java)

The execution result of this sample program is output to the standard output. The following shows the content of the standard output:

```
$ The total is 315000 yen.
```

5.6 Sample Program that Uses the XML Schema

This section describes a sample program that uses an XML Schema.

The sample program is provided only in Windows.

The installation folder and file name of the sample program are as follows.

Installation folder:

installation-folder-of-Cosminexus-XML-Processor\samples\XMLSchema

File name:

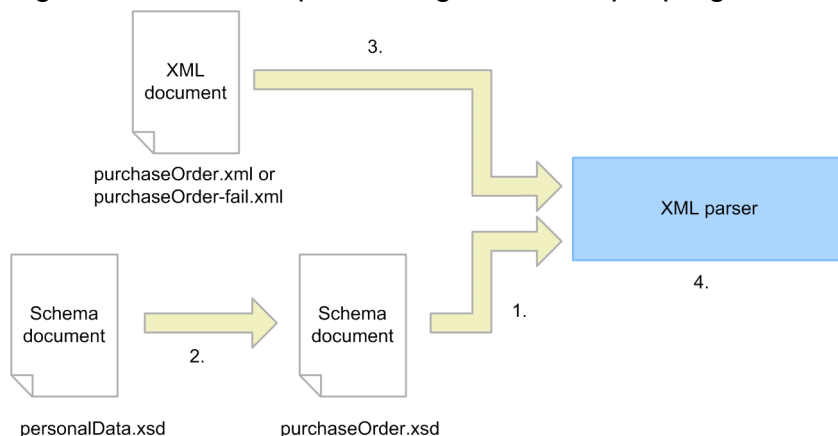
- purchaseOrder.xml
- purchaseOrder-fail.xml
- purchaseOrder.xsd
- personalData.xsd
- SampleValidateDOM.java
- SampleValidateSAX.java

5.6.1 Processing Performed by the Sample Program

The sample program validates a purchase order created as an XML document. Because the purchase order uses the shared customer personal data, the schema document of the purchase order reads the schema document of the personal data. Note that the schema documents are identified in the Java program.

The following figure shows the flow of processing of the sample program that uses the XML Schema.

Figure 5–4: Flow of processing of the sample program that uses the XML schema



The sample program performs the following processing:

1. The program reads the schema document `purchaseOrder.xsd`.
`PurchaseOrder.xsd` is the schema document that defines the product purchase order.
2. `PurchaseOrder.xsd` reads the schema document `personalData.xsd`.

`PersonalData.xsd` is the schema document for defining the personal data of the customers who place an order and has a different namespace than `purchaseOrder.xsd`.

3. The program reads the XML document `purchaseOrder.xml` (or `purchaseOrder-fail.xml`).

`PurchaseOrder.xml` or `purchaseOrder-fail.xml` is the XML document for the purchase order to be validated.

4. The program validates the XML document.

If the XML document is valid against the schema document, the standard output displays `Validation OK`. If not valid, the standard output displays `Validation NG` with a message.

5.6.2 XML Document to Use (`purchaseOrder.xml` and `purchaseOrder-fail.xml`)

Cosminexus XML Processor provides the following types of XML documents for validation by the XML Schema:

1. XML document that is validated successfully (`purchaseOrder.xml`)
2. XML document that fails validation (`purchaseOrder-fail.xml`)

(1) XML document (`purchaseOrder.xml`)

The following shows an XML document (`purchaseOrder.xml`) that is validated successfully.

```
<?xml version="1.0"?>
<po:purchaseOrder
  xmlns:po="http://www.myshopping.com/schema/purchaseOrder"
  xmlns:psd="http://www.myshopping.com/schema/personalData">
  <po:shipTo age="20">
    <psd:firstName>John</psd:firstName>
    <psd:familyName>Doe</psd:familyName>
    <psd:occupation>accountant</psd:occupation>
    <psd:email>johnD@bpl.com</psd:email>
    <psd:tel>050-1234-1234</psd:tel>
    <psd:address country="US">
      <psd:street>Universal street 100</psd:street>
      <psd:city>Carson</psd:city>
      <psd:state>Nevada</psd:state>
      <psd:zip>10-456</psd:zip>
    </psd:address>
  </po:shipTo>
  <po:billTo age="20">
    <psd:firstName>Jane</psd:firstName>
    <psd:familyName>Doe</psd:familyName>
    <psd:occupation>lawyer</psd:occupation>
    <psd:email>janeD@bpl.com</psd:email>
    <psd:tel>033-1111-2345</psd:tel>
    <psd:address country="US">
      <psd:street>Times Square 555</psd:street>
      <psd:city>New York</psd:city>
      <psd:state>New York</psd:state>
      <psd:zip>155-5600</psd:zip>
    </psd:address>
  </po:billTo>
</po:purchaseOrder>
```

```

<po:credit year="2007" month="10">
  <po:creditHolder>Jane Doe</po:creditHolder>
  <po:creditNumber>1111444422229999</po:creditNumber>
  <po:creditCompany>CardCompanyA</po:creditCompany>
</po:credit>
</po:billTo>
<po:items>
  <po:item>
    <po:productName
      productID="DTPC2000S">DeskTop PC</po:productName>
    <po:quantity>1</po:quantity>
    <po:price>1500</po:price>
    <po:shipDate>2004-05-20</po:shipDate>
  </po:item>
  <po:item>
    <po:productName
      productID="DC500MP">Digital Camera</po:productName>
    <po:quantity>1</po:quantity>
    <po:price>450</po:price>
    <po:shipDate>2004-05-20</po:shipDate>
  </po:item>
  <po:item>
    <po:productName
      productID="LP800S">Printer</po:productName>
    <po:quantity>1</po:quantity>
    <po:price>200</po:price>
    <po:shipDate>2004-05-20</po:shipDate>
  </po:item>
</po:items>
</po:purchaseOrder>

```

(2) XML document (purchaseOrder-fail.xml)

The following shows a fragment of an XML document (purchaseOrder-fail.xml) that fails validation.

```

...
...
<po:credit year="2007" month="10">
  <po:creditHolder>Jane Doe</po:creditHolder>
  <po:creditNumber>111144442222999955</po:creditNumber>
  <po:creditCompany>CardCompanyA</po:creditCompany>
</po:credit>
...
...

```

This XML document is identical with the one described in [5.6.2 \(1\) XML document \(purchaseOrder.xml\)](#) except for the definition of the credit number starting with `<po:creditNumber>`. An 18-digit number is used as the value for the credit number starting with `<po:creditNumber>`. However, the sample schema document defines the credit number as 16 digits, resulting in an error at the time of validation.

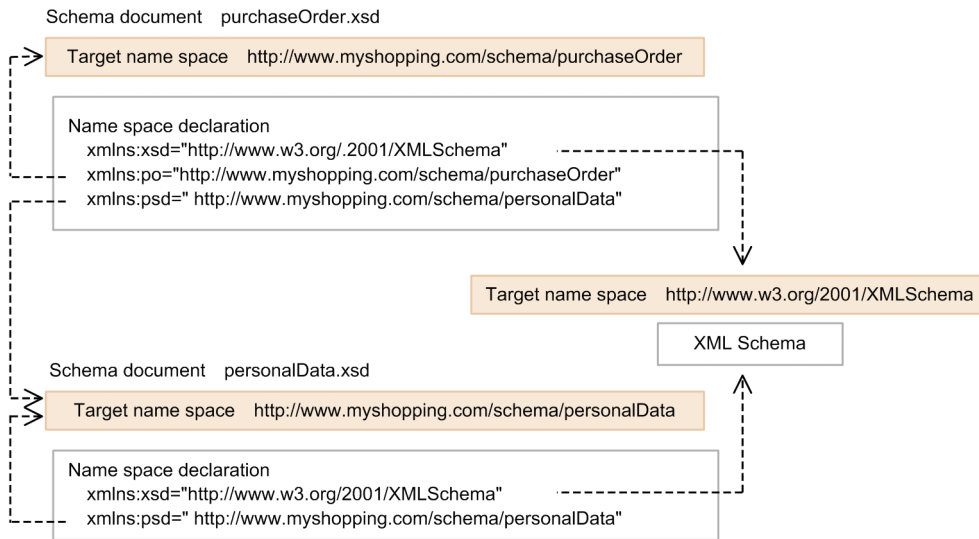
5.6.3 Schema Documents to Use (purchaseOrder.xsd, personalData.xsd)

This sample uses `purchaseOrder.xsd` and `personalData.xsd` as the schema documents.

(1) Relation between namespaces in schema documents

The following figure shows the relation among namespaces in `purchaseOrder.xsd` and `personalData.xsd`. Each arrow indicates the target namespace in which the type definition or element to be referred to is defined.

Figure 5–5: Relation among namespaces in the schema documents



(2) Schema document (purchaseOrder.xsd)

The following shows the schema document to use (`purchaseOrder.xsd`):

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.myshopping.com/schema/purchaseOrder"
  xmlns:po="http://www.myshopping.com/schema/purchaseOrder"
  xmlns:psd="http://www.myshopping.com/schema/personalData">

  <xsd:import
    namespace="http://www.myshopping.com/schema/personalData"
    schemaLocation="personalData.xsd"/>

  <xsd:element name="purchaseOrder" type="po:purchaseOrderType"/>
  <xsd:element name="shipTo" type="psd:personalDataType"/>
  <xsd:element name="billTo" type="po:billToPersonalDataType"/>
  <xsd:element name="items" type="po:itemsType">
    <xsd:key name="productID_unique">
      <xsd:selector xpath="po:item/po:productName"/>
      <xsd:field xpath="@productID"/>
    </xsd:key>
  </xsd:element>

  <xsd:element name="item" type="po:itemType"/>
  <xsd:element name="productName" type="po:productNameType"/>
  <xsd:element name="quantity">
    <xsd:simpleType>
      <xsd:restriction base="xsd:positiveInteger">
        <xsd:maxExclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
</xsd:schema>
```

```

</xsd:element>

<xsd:element name="price" type="xsd:positiveInteger"/>
<xsd:element name="shipDate" type="xsd:date"/>
<xsd:element name="credit" type="po:creditType"/>
<xsd:element name="creditHolder" type="xsd:string"/>
<xsd:element name="creditNumber">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{16}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="creditCompany">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="CardCompanyA"/>
      <xsd:enumeration value="CardCompanyB"/>
      <xsd:enumeration value="CardCompanyC"/>
      <xsd:enumeration value="CardCompanyD"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:complexType name="purchaseOrderType">
  <xsd:sequence>
    <xsd:element ref="po:shipTo"/>
    <xsd:element ref="po:billTo"/>
    <xsd:element ref="po:items"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="billToPersonalDataType">
  <xsd:sequence>
    <xsd:group ref="psd:personalDataGroup"/>
    <xsd:element ref="po:credit"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="psd:personalAttributeGroup"/>
</xsd:complexType>

<xsd:complexType name="itemsType">
  <xsd:sequence>
    <xsd:element ref="po:item"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="itemType">
  <xsd:sequence>
    <xsd:element ref="po:productName"/>
    <xsd:element ref="po:quantity"/>
    <xsd:element ref="po:price"/>
    <xsd:element ref="po:shipDate"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="productNameType">

```

```

    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="productID" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="creditType">
    <xsd:sequence>
      <xsd:element ref="po:creditHolder"/>
      <xsd:element ref="po:creditNumber"/>
      <xsd:element ref="po:creditCompany"/>
    </xsd:sequence>
    <xsd:attribute name="year" type="po:yearType" use="required"/>
    <xsd:attribute name="month" type="po:monthType" use="required"/>
  </xsd:complexType>

  <xsd:simpleType name="yearType">
    <xsd:annotation>
      <xsd:documentation>
        Year set from 2004 to 2008
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="200[4-8]"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="monthType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="0[1-9]|1[0-2]"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>

```

(3) Schema document (personalData.xsd)

The following shows the schema document to use (personalData.xsd):

```

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.myshopping.com/schema/personalData"
  xmlns:psd="http://www.myshopping.com/schema/personalData">

  <xsd:element name="personalData" type="psd:personalDataType"/>
  <xsd:element name="firstName" type="xsd:string"/>
  <xsd:element name="familyName" type="xsd:string"/>
  <xsd:element name="occupation" type="xsd:string"/>
  <xsd:element name="email" type="xsd:string"/>
  <xsd:element name="tel" type="xsd:string"/>
  <xsd:element name="address" type="psd:addressType"/>
  <xsd:element name="building" type="xsd:string"/>
  <xsd:element name="street" type="xsd:string"/>
  <xsd:element name="city" type="xsd:string"/>
  <xsd:element name="state" type="xsd:string"/>
  <xsd:element name="zip" type="xsd:string"/>

```



```

<xsd:complexType name="personalDataType">
  <xsd:group ref="psd:personalDataGroup"/>
  <xsd:attributeGroup ref="psd:personalAttributeGroup"/>
</xsd:complexType>

<xsd:group name="personalDataGroup">
  <xsd:sequence>
    <xsd:element ref="psd:firstName"/>
    <xsd:element ref="psd:familyName"/>
    <xsd:element ref="psd:occupation"/>
    <xsd:element ref="psd:email"/>
    <xsd:element ref="psd:tel"/>
    <xsd:element ref="psd:address"/>
  </xsd:sequence>
</xsd:group>

<xsd:attributeGroup name="personalAttributeGroup">
  <xsd:attribute name="age"
    type="xsd:positiveInteger" use="optional"/>
  <xsd:attribute name="company"
    type="xsd:string" use="optional"/>
  <xsd:attribute name="department"
    type="xsd:string" use="optional"/>
  <xsd:attribute name="title"
    type="xsd:string" use="optional"/>
  <xsd:attribute name="fax"
    type="xsd:string" use="optional"/>
</xsd:attributeGroup>

<xsd:complexType name="addressType">
  <xsd:sequence>
    <xsd:element ref="psd:building" minOccurs="0"/>
    <xsd:element ref="psd:street"/>
    <xsd:element ref="psd:city"/>
    <xsd:element ref="psd:state"/>
    <xsd:element ref="psd:zip"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:string"
    use="optional" default="US"/>
</xsd:complexType>
</xsd:schema>

```

5.6.4 Sample Program When Using the DOM parser

This section describes a sample program for validating XML documents using the DOM parser, how the sample program is executed, and the execution results.

(1) Sample program (SampleValidateDOM.java)

The following shows the sample program (SampleValidateDOM.java) when using the DOM parser:

```

import javax.xml.parsers.*;
import org.w3c.dom.*;

```

```

import org.xml.sax.*;
import java.io.*;

public class SampleValidateDOM implements ErrorHandler{
    public static final void main(String[] argv){
        if (argv.length != 2) {
            System.out.println(
                "Usage: java SampleValidateDOM <xml_file> <schema_file>");
            System.exit(1);
        }

        try{
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            dbf.setNamespaceAware(true);
            dbf.setValidating(true);
            dbf.setAttribute(
                "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
                "http://www.w3.org/2001/XMLSchema");
            dbf.setAttribute(
                "http://java.sun.com/xml/jaxp/properties/schemaSource",
                argv[1]);

            DocumentBuilder db = dbf.newDocumentBuilder();
            db.setErrorHandler(new SampleValidateDOM());
            Document doc = db.parse(argv[0]);
            System.out.println("Validation OK");

        }catch(ParserConfigurationException e){
            e.printStackTrace();
        }catch(SAXParseException e){
            e.printStackTrace();
        }catch(SAXException e){
            System.out.println("Validation NG: " + e.getMessage());
        }catch(IOException e){
            e.printStackTrace();
        }
    }

    public void warning(SAXParseException exception)
        throws SAXException {
        System.out.println("***Parsing Warning**\n" +
            "  Line:      " +
            exception.getLineNumber() + "\n" +
            "  URI:        " +
            exception.getSystemId() + "\n" +
            "  Message: " +
            exception.getMessage());
    }

    public void error(SAXParseException exception)
        throws SAXException {
        System.out.println("***Parsing Error**\n" +
            "  Line:      " +
            exception.getLineNumber() + "\n" +
            "  URI:        " +
            exception.getSystemId() + "\n" +
            "  Message: " +
            exception.getMessage());
    }
}

```

```

        exception.getMessage());
    throw new SAXException("Error encountered");
}

public void fatalError(SAXParseException exception)
    throws SAXException {
    System.out.println("**Parsing Fatal Error**\n" +
        "    Line:      " +
        exception.getLineNumber() + "\n" +
        "    URI:        " +
        exception.getSystemId() + "\n" +
        "    Message:    " +
        exception.getMessage());
    throw new SAXException("Fatal Error encountered");
}
}

```

(2) Execution result of the sample program

The execution result of this sample program is output to the standard output. The following shows the content of the standard output:

When `purchaseOrder.xml` is specified for validation

```
Validation OK
```

When `purchaseOrder-fail.xml` is specified for validation

```

**Parsing Error**
Line:      33
URI:       file:///folder-where-sample-is-stored/purchaseOrder-fail.xml
Message:   KECX06063-E cvc-pattern-valid: Value '111144442222999955' is no
t facet-valid with respect to pattern '\d{16}' for type '#AnonType_creditN
umber'.
Validation NG: Error encountered

```

5.6.5 Sample Program When Using the SAX Parser

This section describes a sample program for validating XML documents using the SAX parser, how the sample program is executed, and the execution result.

(1) Sample program (SampleValidateSAX.java)

The following shows the sample program (`SampleValidateSAX.java`) when using the SAX parser:

```

import javax.xml.parsers.*;
import org.xml.sax.*;
import java.io.*;

public class SampleValidateSAX implements ErrorHandler{
    public static final void main(String[] argv){
        if (argv.length != 2) {
            System.out.println(
                "Usage: java SampleValidateSAX <xml_file> <schema_file>");

```

```

        System.exit(1);
    }

    try{
        SAXParserFactory spf = SAXParserFactory.newInstance();
        spf.setNamespaceAware(true);
        spf.setValidating(true);
        SAXParser sp = spf.newSAXParser();
        sp.setProperty(
            "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
            "http://www.w3.org/2001/XMLSchema");
        sp.setProperty(
            "http://java.sun.com/xml/jaxp/properties/schemaSource",
            argv[1]);

        XMLReader reader = sp.getXMLReader();
        reader.setErrorHandler(new SampleValidateSAX());
        reader.parse(argv[0]);
        System.out.println("Validation OK");

    }catch(ParserConfigurationException e){
        e.printStackTrace();
    }catch(SAXParseException e){
        e.printStackTrace();
    }catch(SAXException e){
        System.out.println("Validation NG: " + e.getMessage());
    }catch(IOException e){
        e.printStackTrace();
    }
}

public void warning(SAXParseException exception)
    throws SAXException {
    System.out.println("***Parsing Warning**\n" +
        "   Line:      " +
        exception.getLineNumber() + "\n" +
        "   URI:        " +
        exception.getSystemId() + "\n" +
        "   Message: " +
        exception.getMessage());
}

public void error(SAXParseException exception)
    throws SAXException{
    System.out.println("***Parsing Error**\n" +
        "   Line:      " +
        exception.getLineNumber() + "\n" +
        "   URI:        " +
        exception.getSystemId() + "\n" +
        "   Message: " +
        exception.getMessage());
    throw new SAXException("Error encountered");
}

public void fatalError(SAXParseException exception)
    throws SAXException {
    System.out.println("***Parsing Fatal Error**\n" +
        "   Line:      " +

```

```

        exception.getLineNumber() + "\n" +
        "    URI:      " +
        exception.getSystemId() + "\n" +
        "    Message: " +
        exception.getMessage());
    throw new SAXException("Fatal Error encountered");
}
}

```

(2) Execution result of the sample program

The execution result of this sample program is output to the standard output. The following shows the content of the standard output:

When `purchaseOrder.xml` is specified for validation

```
Validation OK
```

When `purchaseOrder-fail.xml` is specified for validation

```

**Parsing Error**
Line:      33
URI:       file:///folder-where-sample-is-stored/purchaseOrder-fail.xml
Message:   KECX06063-E cvc-pattern-valid: Value '111144442222999955' is no
t facet-valid with respect to pattern '\d{16}' for type '#AnonType_creditN
umber'.
Validation NG: Error encountered

```

5.7 Sample Program that Uses the XSLT Transformer

This section describes a sample program that uses the XSLT transformer.

The sample program is provided only in Windows.

The installation folder and file name of the sample program are as follows.

Installation folder:

installation-folder-of-Cosminexus-XML-Processor\samples\XSLT

File name:

- SampleXSLT.xml
- SampleXSLT.xsl
- SampleXSLT.java

5.7.1 Processing Performed by the Sample Program

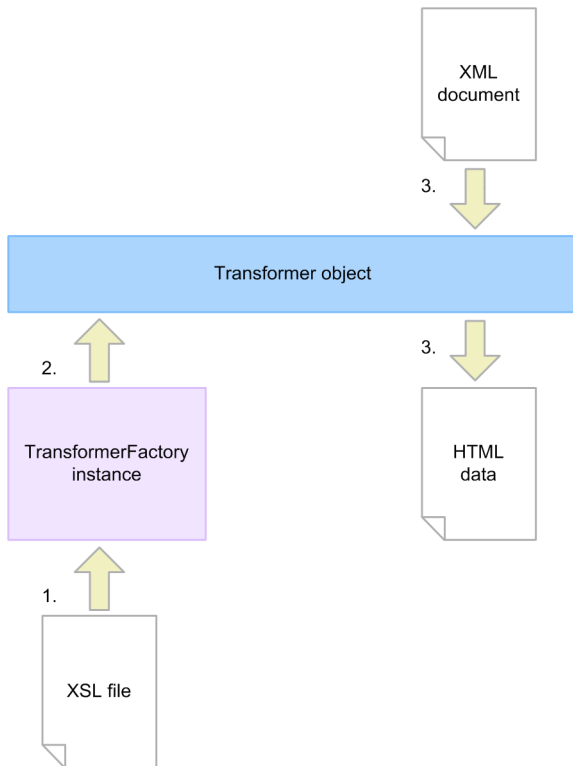
The sample program performs the following processing:

1. The program reads the source XML document.
2. Next, the program transforms the XML document read in step1. into HTML data.

5.7.2 General Procedure for Creating the Sample Program

The following figure shows the general procedure for creating a sample program that uses the XSLT transformer.

Figure 5–6: General procedure for creating a sample program that uses the XSLT transformer



1. Get the `TransformerFactory` instance.

Get the `TransformerFactory` instance using the `newInstance` method.

2. Create a `Transformer` object.

3. Transform into HTML data.

Transform the XML document into HTML data by using the `Transformer` object.

5.7.3 XML Document to Use (SampleXSLT.xml)

The following shows the XML document read by the sample program.

```
<?xml version="1.0" encoding="Shift_JIS"?>
<?xml-stylesheet type="text/xsl" href="SampleXSLT.xsl" ?>
<START>
  <Payment_Slip>
    <Merchandise>
      <Brand_Name>Television</Brand_Name>
      <Unit_Price>15000</Unit_Price>
      <Amount>14</Amount>
    </Merchandise>
    <Merchandise>
      <Brand_Name>Radio</Brand_Name>
      <Unit_Price>3500</Unit_Price>
      <Amount>6</Amount>
    </Merchandise>
    <Merchandise>
      <Brand_Name>Video</Brand_Name>
```

```

        <Unit_Price>21000</Unit_Price>
        <Amount>4</Amount>
    </Merchandise>
</Payment_Slip>
</START>

```

5.7.4 XSL File to Use (SampleXSLT.xsl)

The following shows the XSL file read by the sample program.

```

<?xml version="1.0" encoding="Shift_JIS"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="html" indent="yes" encoding="Shift_JIS"/>

    <xsl:template match="/">
        <xsl:apply-templates />
    </xsl:template>

    <!-- start conversion for START as a root -->
    <xsl:template match="START">
        <HTML>
            <HEAD>
                <TITLE>Display XML at a table of HTML</TITLE>
            </HEAD>
            <BODY BGCOLOR="#C0C0C0">
                <FONT SIZE="5">Ordering Payment Slip</FONT>
                <TABLE BORDER="2">
                    <TR><TH>Brand_Name</TH><TH>Unit_Price</TH>
                        <TH>Amount</TH><TH>Subtotal</TH></TR>
                    <xsl:apply-templates />
                    <TR><TH>Total</TH><TD>&#160;</TD><TD>&#160;</TD>
                        <TD ALIGN="RIGHT"><xsl:value-of
                            select="(//Merchandise[1]/Unit_Price * //Merchandise[1]/Amount +
                                //Merchandise[2]/Unit_Price * //Merchandise[2]/Amount +
                                //Merchandise[3]/Unit_Price * //Merchandise[3]/Amount)" /
                        ></TD></TR>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:template>

    <xsl:template match="Payment_Slip">
        <xsl:apply-templates />
    </xsl:template>

    <!-- make one line with one merchandise, and output a subtotal -->
    <xsl:template match="Merchandise">
        <TR><xsl:apply-templates />
        <TD ALIGN="RIGHT">
            <xsl:value-of select="number(Unit_Price)*number(Amount)" />
        </TD></TR>
    </xsl:template>

    <!-- output a brand name -->

```



```

<xsl:template match="Brand_Name">
  <TD><xsl:value-of select="."/></TD>
</xsl:template>

<!-- output unit price -->
<xsl:template match="Unit_Price">
  <TD ALIGN="RIGHT"><xsl:value-of select="."/></TD>
</xsl:template>

<!-- output amount -->
<xsl:template match="Amount">
  <TD ALIGN="RIGHT"><xsl:value-of select="."/></TD>
</xsl:template>
</xsl:stylesheet>

```

5.7.5 Sample Program (SampleXSLT.java)

The following shows the sample program:

```

import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.parsers.*;

public class SampleXSLT{
  public static void main(String[] args){

    try{
      File file = new File(args[0]);

      //read XML file
      Source source = new StreamSource(file);
      Result result = new StreamResult(System.out);
      TransformerFactory factory =
        TransformerFactory.newInstance();

      //convert to HTML file
      Source style = factory.getAssociatedStylesheet(source,
        null, null, null);

      Transformer transformer = factory.newTransformer(style);
      transformer.transform(source, result);

    }catch(TransformerConfigurationException e){
      e.printStackTrace();
    }catch(TransformerException e){
      e.printStackTrace();
    }
  }
}

```

5.7.6 Execution Result of the Sample Program (SampleXSLT.java)

(1) Execution result of the sample program

The execution result of this sample program is output to the standard output. The following shows the content of the standard output:

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<TITLE>Display XML at a table of HTML</TITLE>
</HEAD>
<BODY BGCOLOR="#C0C0C0">
<FONT SIZE="5">Ordering Payment Slip</FONT>
<TABLE BORDER="2">
<TR>
<TH>Brand_Name</TH><TH>Unit_Price</TH><TH>Amount</TH><TH>Subtotal</TH>
</TR>
<TR>
<TD>Television</TD>
      <TD ALIGN="RIGHT">15000</TD>
      <TD ALIGN="RIGHT">14</TD>
      <TD ALIGN="RIGHT">210000</TD>
</TR>
<TR>
<TD>Radio</TD>
      <TD ALIGN="RIGHT">3500</TD>
      <TD ALIGN="RIGHT">6</TD>
      <TD ALIGN="RIGHT">21000</TD>
</TR>
<TR>
<TD>Video</TD>
      <TD ALIGN="RIGHT">21000</TD>
      <TD ALIGN="RIGHT">4</TD>
      <TD ALIGN="RIGHT">84000</TD>
</TR>
<TR>
<TH>Total</TH><TD>&nbsp;</TD><TD>&nbsp;</TD><TD ALIGN="RIGHT">315000</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

5.8 Sample Program that Uses the XSLTC Transformer

This section describes a program that uses the XSLTC transformer.

The section provides a method for using the XSLTC transformer function by changing the sample program described in [5.7 Sample Program that Uses the XSLT Transformer](#).

The following figure shows the changes to the program to enable use of the XSLTC transformer function. The changes shown in this figure make the XSLTC transformer function available. The method for executing the program and the execution result are identical to the sample program for using XSLT transformers. For details, see [5.7.6 Execution Result of the Sample Program \(SampleXSLT.java\)](#).

Important note

Do not change the file name even if you change the program. Compilation fails if the file name is changed.

Figure 5–7: Changes to the program to enable use of the XSLTC transformer function

(Before change)	(After change)
<pre>import java.io.*; import javax.xml.transform.*; import javax.xml.transform.stream.*; import javax.xml.parsers.*; public class SampleXSLT{ public static void main(String[] args){ try{ File file = new File(args[0]); //Read XML file Source source = new StreamSource(file); Result result = new StreamResult(System.out); TransformerFactory factory = TransformerFactory.newInstance(); //Transformation to HTML file Source style = factory.getAssociatedStylesheet(source, null, null, null); Transformer transformer = factory.newTransformer(style); transformer.transform(source, result); }catch(TransformerConfigurationException e){ e.printStackTrace(); }catch(TransformerException e){ e.printStackTrace(); } } }</pre>	<pre>import java.io.*; import javax.xml.transform.*; import javax.xml.transform.stream.*; import javax.xml.parsers.*; import com.cosminexus.jaxp.xsltc.*; //...1 public class SampleXSLT{ public static void main(String[] args){ try{ File file = new File(args[0]); //Read XML file Source source = new StreamSource(file); Result result = new StreamResult(System.out); TransformerFactory factory = TransformerFactoryXSLTC.newInstance(); //...2 //Transformation to HTML file Source style = factory.getAssociatedStylesheet(source, null, null, null); Transformer transformer = factory.newTransformer(style); transformer.transform(source, result); }catch(TransformerConfigurationException e){ e.printStackTrace(); }catch(TransformerException e){ e.printStackTrace(); } } }</pre>

The following describes the changes to the program. Each item number corresponds to the number in the comments in the program.

1. Import the package to use the XSLTC transformer function.
2. Apply the `newInstance` method to the `TransformerFactoryXSLTC` class and generate the `TransformerFactory` object.

6

Notes on Using Cosminexus XML Processor

This chapter gives cautionary notes on using Cosminexus XML Processor.

6.1 Notes common to JAXP1.4 functions

The following table gives general notes on JAXP 1.4:

Table 6–1: General notes on JAXP 1.4

No.	Notes
1	For some methods in the JAXP 1.4 specification, the behavior when an argument is set to null is not defined. Note that if you specify null for an argument of these methods, they might result in undefined behavior, including a <code>NullPointerException</code> .
2	If an exception occurs in one of the methods defined in the JAXP 1.4 specification, applying the <code>getMessage</code> method to the exception object can often provide detailed messages. However, you cannot always get detailed messages. Therefore, in the application, proper processing must be executed according to the exception type instead of the exception handling dependent on the content of detailed messages. For example, in case of a <code>DOMException</code> , determine the detailed cause of the error by using the <code>code</code> field.
3	Specify the correct format for URIs to be specified in the arguments of the <code>parse(String uri)</code> method of the <code>DocumentBuilder</code> class and <code>parse(String uri, ...)</code> method of the <code>SAXParser</code> class. If you specify an invalid string as URI, a <code>RuntimeException</code> exception, such as <code>IllegalArgumentException</code> might occur.
4	The <code>File</code> object to be specified in the arguments of the <code>parse(File f)</code> method of the <code>DocumentBuilder</code> class and <code>parse(File f, ...)</code> method of the <code>SAXParser</code> class must be generated by a <code>File</code> constructor that does not include a percent sign (%), hash mark (#), or question mark (?) in the arguments. If you specify a <code>File</code> object generated by the <code>File</code> constructor that contains these characters in its arguments, the following phenomena might occur: <ul style="list-style-type: none">• Occurrence of a <code>RuntimeException</code> exception, such as <code>IllegalArgumentException</code>• Occurrence of a <code>SAXException</code> exception• The operations of the XML parser are invalid
5	Unify the encoding of XML documents, encoding of the <code>pseudo</code> attribute of the XML declaration, and encoding of <code>InputSource</code> and <code>Reader</code> . Only operations with unified encoding are guaranteed.

When you use JAXP1.4, you cannot use the classes and methods listed in the following table.

Table 6–2: Classes and methods not supported in JAXP1.4

No.	Package name	Class name	Method name
1	javax.xml.stream	XMLEventFactory	<code>newInstance(String, ClassLoader)</code>
2		XMLInputFactory	<code>newInstance(String, ClassLoader)</code>
3		XMLOutputFactory	<code>newInstance(String, ClassLoader)</code>

Also, you can only specify the factory class names described in the table for the methods listed in the following table. The operations might not function normally, if other factory class names are specified.

Table 6–3: Factory-generating methods and specifiable factory class names

No.	Package name	Class name	Method name	Specifiable factory class name
1	javax.xml.datatype	<code>DatatypeFactory</code>	<code>newInstance(String, ClassLoader)</code>	<code>com.cosminexus.jaxp.impl.parsers.jaxp.datatype.DatatypeFactoryImpl</code>
2	javax.xml.parsers	<code>DocumentBuilderFactory</code>	<code>newInstance(String, ClassLoader)</code>	<code>com.cosminexus.jaxp.impl.parsers.jaxp.DocumentBuilderFactoryImpl</code>
3		<code>SAXParserFactory</code>	<code>newInstance(String, ClassLoader)</code>	<code>com.cosminexus.jaxp.impl.parsers.jaxp.SAXParserFactoryImpl</code>

No.	Package name	Class name	Method name	Specifiable factory class name
4	javax.xml.stream	XMLEventFactory	newFactory(String, ClassLoader)	com.cosminexus.stax.xml.stream.events.ZephyrEventFactory
5				javax.xml.stream.XMLEventFactory#
6		XMLInputFactory	newFactory(String, ClassLoader)	com.cosminexus.stax.xml.stream.ZephyrParserFactory
7				javax.xml.stream.XMLInputFactory#
8		XMLOutputFactory	newFactory(String, ClassLoader)	com.cosminexus.stax.xml.stream.ZephyrWriterFactory
9				javax.xml.stream.XMLOutputFactory#
10	javax.xml.transform	TransformerFactory	newInstance(String, ClassLoader)	com.cosminexus.jaxp.impl.transform.processor.TransformerFactoryImpl
11	javax.xml.validation	SchemaFactory	newInstance(String, String, ClassLoader)	com.cosminexus.jaxp.impl.parsers.jaxp.validation.XMLSchemaFactory
12	javax.xml.xpath	XPathFactory	newInstance(String, String, ClassLoader)	com.cosminexus.jaxp.impl.xpath.jaxp.XPathFactoryImpl

#

Supported in Developer's Kit for Java 09-60 or later

6.1.1 Range supported in the javax.xml.transform.stax.StAXSource class and the javax.xml.transform.stax.StAXResult class

In version 08-50 and later versions of Cosminexus XML Processor, you can use the javax.xml.transform.stax.StAXSource class and javax.xml.transform.stax.StAXResult class.

Some of the methods defined in JAXP 1.4 and JAXB use the javax.xml.transform.stax.StAXSource class and the javax.xml.transform.stax.StAXResult class, or the FEATURE field of these classes as arguments. However, some of the methods are not supported. The following table describes the status of support.

Table 6–4: Status of support for methods using StAXSource class or StAXResult class in arguments

No.	Package name	Class name	Method name	Support
1	javax.xml.transform	TransformerFactory	getAssociatedStylesheet(Source, String, String, String)	N
2			getFeature(String)	N
3			newTemplates(Source)	Y
4			newTransformer(Source)	Y

No.	Package name	Class name	Method name	Support
5		Transformer	transform(Source, Result)	Y
6	javax.xml.transform.sax	SAXTransformerFactory	newTransformerHandler(Source)	N
7			newXMLFilter(Source)	Y
8		SAXSource	sourceToInputSource(Source)	N
9		TransformerHandler	setResult(Result)	Y
10	javax.xml.validation	SchemaFactory	newSchema(Source)	N
11			newSchema(Source[])	N
12		Validator	validate(Source)	Y
13			validate(Source, Result)	Y
14	javax.xml.bind	JAXB	marshal(Object, Result)	N
15			unmarshal(Source, Class<T>)	N
16		Marshaller	marshal(Object, Result)	N
17		Unmarshaller	unmarshal(Source)	N
18			unmarshal(Source, Class<T>)	N
19	javax.xml.stream	XMLOutputFactory	createXMLEventWriter(Result)	Y
20			createXMLStreamWriter(Result)	Y
21		XMLInputFactory	createXMLEventReader(Source)	Y
22			createXMLStreamReader(Source)	Y

Legend

Y: Supported.

N: Not supported.

6.1.2 Other notes

The following table describes notes other than those described for JAXP 1.4 in the previous sections.

Reference note

For related notes, see [6.4 Notes on StAX](#) as and when required.

Table 6–5: Notes on using JAXP 1.4

No.	Notes
1	StAX does not support the Shift_JIS switch functionality. The operations might not function properly if the Node, Source, and Result created by a DOM parser or a SAX parser, specifying the Shift_JIS switch functionality, are passed as arguments of a StAX API.
2	StAX does not support the Shift_JIS switch functionality. The operations cannot be guaranteed if StAXSource or StAXResult is used in the I/O of the XSLT or the XSLTC transformer.
3	The following features have been added to XSLT, but Cosminexus XML Processor always returns false:

No.	Notes
	<ul style="list-style-type: none"> • <i>http://javax.xml.transform.stax.StAXSource/feature</i> • <i>http://javax.xml.transform.stax.StAXResult/feature</i>

6.2 Notes on the DOM Parser

The following table gives cautionary notes on the DOM parser.

Table 6–6: Notes on the DOM parser

No.	Notes
1	<p>In multi-thread programming, the <code>DocumentBuilderFactory</code> class is not thread-safe. Therefore, multiple threads must not access the same <code>DocumentBuilderFactory</code> instance at the same time. To avoid conflicts between threads, use one of the following methods:</p> <ul style="list-style-type: none">• Each thread has one <code>DocumentBuilderFactory</code> instance.• Each thread exclusively accesses the <code>DocumentBuilderFactory</code> instance.
2	<p>In multi-thread programming, the <code>DocumentBuilder</code> class is not thread-safe. Therefore, multiple threads must not use the same <code>DocumentBuilder</code> instance at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread has one <code>DocumentBuilder</code> instance.
3	<p>In multi-thread programming, the DOM tree is not thread-safe. Therefore, multiple threads must not access the same DOM tree generated by the <code>parse</code> method at the same time. Not only update methods, but also reference methods must not access such trees at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread exclusively accesses the DOM tree.
4	<p>In multi-thread programming, the objects defined by <code>org.w3c.dom</code>, <code>org.w3c.dom.bootstrap</code>, and <code>org.w3c.dom.ls</code> packages are not thread-safe. Therefore, multiple threads must not access these objects at the same time. Not only update methods, but also reference methods must not access such trees at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread exclusively accesses these objects.
5	<p>If you generate an <code>Attr</code> node by using the <code>createAttribute</code> or <code>createAttributeNS</code> method of the <code>Document</code> interface, set a value in the <code>setValue</code> method of the <code>Attr</code> interface. If you get <code>NodeList</code> in the <code>getChildNodes</code> method of the <code>Node</code> interface while the value is not set, the <code>NodeList</code>'s behavior is not guaranteed.</p>
6	<p>If you add an element to the <code>Document</code> node that already has an <code>Element</code> node by using the <code>insertBefore</code> or the <code>appendChild</code> method of the <code>Node</code> interface, the result is the same as when you use the <code>replaceChild</code> method of the <code>Node</code> interface.</p>
7	<p>When an error occurs in the parse methods that take <code>InputStream</code> or <code>InputSource</code> as an argument of the <code>DocumentBuilder</code> class, null might be returned if you apply the <code>getSystemId</code> method to the <code>SAXParseException</code> passed to the error handler. If you want to return the system identifier of the error source, use the <code>parse</code> method as follows:</p> <ul style="list-style-type: none">• In the <code>parse(InputStream is, String systemId)</code> method, specify the system identifier for the <code>systemId</code> argument.• In the <code>parse(InputSource is)</code> method, specify the <code>InputSource</code> that has the system identifier set for the <code>is</code> argument.
8	<p>If you analyze an XML document saved in UTF-16 with BOM (Byte Order Mark) by using the <code>parse(InputSource is)</code> method, specify UTF-16 for the argument if you apply the <code>setEncoding</code> method to the <code>InputSource</code>.</p>
9	<p>If an element name with 477 or more characters is included in the character string that defines an internal entity, the <code>java.lang.IndexOutOfBoundsException</code> exception might occur.</p>

6.3 Notes on the SAX Parser

The following table gives cautionary notes on the SAX parser.

Table 6–7: Notes on the SAX parser

No.	Notes
1	<p>In multi-thread programming, the <code>SAXParserFactory</code> class is not thread-safe. Therefore, multiple threads must not access the same <code>SAXParserFactory</code> instance at the same time. To avoid conflicts between threads, use one of the following methods:</p> <ul style="list-style-type: none">• Each thread has one <code>SAXParserFactory</code> instance.• Each thread exclusively accesses the <code>SAXParserFactory</code> instance.
2	<p>In multi-thread programming, the <code>SAXParser</code> class is not thread-safe. Therefore, multiple threads must not use the same <code>SAXParser</code> instance at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread has one <code>SAXParser</code> instance.
3	<p>In multi-thread programming, the objects defined by <code>org.xml.sax</code>, <code>org.xml.sax.ext</code>, and <code>org.xml.sax.helpers</code> packages are not thread-safe. Therefore, multiple threads must not access these objects at the same time. Not only update methods, but also reference methods must not access such trees at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread exclusively accesses these objects.
4	<p>The SAX parser can divide single character string data into multiple character strings (chunks) and report to the application as multiple characters events. Therefore, in the <code>ContentHandler</code> implementation, you must be concerned that character string data might be divided.</p> <p>An example of the <code>ContentHandler</code> implementation is coded below. In this example, the character string data is buffered each time a characters event occurs, and the character string data is considered to be ended when an <code>endElement</code> event is reported.</p> <pre>class MyHandler implements ContentHandler { String str = null; StringBuffer buffer = null; : public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException { buffer = new StringBuffer(); } : public void characters(char c[], int start, int length) throws SAXException { buffer.append(c, start, length); } : public void endElement(String uri, String name, String qname) throws SAXException { str = buffer.toString(); } : }</pre>
5	<p>If you generate an <code>XMLReader</code> by using the <code>createXMLReader(String className)</code> method of the <code>XMLReaderFactory</code> class, specify <code>com.cosminexus.jaxp.impl.parsers.parsers.SAXParser</code> for the <code>className</code> argument. An example of <code>XMLReader</code> generation is as follows:</p> <pre>XMLReader reader = XMLReaderFactory.createXMLReader("com.cosminexus.jaxp.impl.parsers.parsers.SAXParser") ;</pre>
6	<p>When an error occurs in the parse methods that take <code>InputStream</code> or <code>InputSource</code> as an argument of the SAX parser class, null might be returned if you apply the <code>getSystemId</code> method to the <code>SAXParseException</code> passed to the error handler. If you want to return the system identifier of the error source, use the parse method as follows:</p> <ul style="list-style-type: none">• In the <code>parse(InputStream is, ..., String systemId)</code> method, specify the system identifier for the <code>systemId</code> argument.

No.	Notes
	<ul style="list-style-type: none"> In the <code>parse (InputSource is, ...)</code> method, specify the <code>InputSource</code> that has the system identifier set for the <code>is</code> argument.
7	When you parse an XML document saved in UTF-16 with BOM (Byte Order Mark) by using the <code>parse (InputSource is ...)</code> method, specify UTF-16 for the argument if you apply the <code>setEncoding</code> method to the <code>InputSource</code> .
8	If an element name with 477 or more characters is included in the character string that defines an internal entity, the <code>java.lang.IndexOutOfBoundsException</code> exception might occur.
9	<p>The <code>java.lang.IllegalStateException</code> exception might occur, if the following conditions are satisfied:</p> <ol style="list-style-type: none"> The <code>TypeInfoProvider</code> object is acquired with the <code>ValidatorHandler.getTypeInfoProvider</code> method. The <code>getElementTypeInfo</code> method is invoked for the object described in step 1 with the event handler of the <code>endElement</code> event of SAX.

6.4 Notes on StAX

The following table lists the notes on StAX.

Table 6–8: Notes on StAX

No.	Notes
1	<p>The StAX 1.0 specifications include methods for which operations are not clearly defined. The operations of such methods depend on the implementation. Check the arguments beforehand so that there are no undefined operations.</p> <p>Examples of undefined operations are as follows:</p> <p>Example 1: An exception of a type that is not described in the Javadoc exception column occurs.</p> <p>The <code>NullPointerException</code> occurs because the operation to be performed when null is specified in an argument is not defined.</p> <p>Example 2: An exception occurs because the operations to be performed when methods are invoked in an invalid order are not defined.</p> <p>An <code>XMLStreamException</code> occurs if you invoke the <code>writeEndDocument</code> method before invoking the <code>writeStartDocument</code> method for the <code>XMLStreamWriter</code> interface.</p> <p>Example 3: If a method is invoked with arguments that might cause a conflict, the arguments are ignored.</p> <p>If you specify different element names as arguments in the <code>createStartElement</code> and <code>createEndElement</code> methods for the <code>XMLEventFactory</code> interface, the argument specified in the <code>createEndElement</code> method is ignored.</p>
2	<p>Some methods of the <code>XMLStreamReader</code> interface have a different description in Javadoc and APIs. Cosminexus XML Processor operates as per the API description.</p> <p>The methods that have different descriptions and differences are as follows:</p> <p>The <code>IllegalStateException</code> exception does not occur (not described in Javadoc)</p> <ul style="list-style-type: none">• <code>getCharacterEncodingScheme()</code>• <code>getEncoding()</code>• <code>getVersion()</code>• <code>getNamespaceURI()</code>• <code>getNamespaceURI(String prefix)</code>• <code>getPrefix()</code>• <code>standaloneSet()</code>• <code>isStandalone()</code> <p>The thrown exception differs from that in Javadoc (the <code>IllegalStateException</code> exception occurs in Javadoc)</p> <ul style="list-style-type: none">• <code>getElementText()</code>• <code>nextTag()</code>• <code>next()</code>
3	<p>Condition</p> <p>When you specify an XML document with different <code>systemId</code> and <code>stream</code> arguments for the <code>createXMLStreamReader(String systemId, InputStream stream)</code> method of <code>XMLInputFactory</code></p> <p>Standard specifications</p> <p>No operations are defined.</p> <p>Operations of Cosminexus XML Processor</p> <p>The argument <code>stream</code> is enabled.</p> <p>However, if the argument <code>stream</code> is null, the argument <code>systemId</code> is enabled.</p>
4	<p>Condition</p> <p>When you specify an XML document with different <code>systemId</code> and <code>reader</code> arguments for the <code>createXMLStreamReader(String systemId, Reader reader)</code> method of <code>XMLInputFactory</code></p> <p>Standard specifications</p> <p>No operations are defined.</p> <p>Operations of Cosminexus XML Processor</p> <p>The argument <code>reader</code> is enabled.</p>

No.	Notes
	However, if the argument <code>reader</code> is null, the argument <code>systemId</code> is enabled.
5	<p>Condition</p> <p>When you execute the <code>getEventType()</code> method of the <code>javax.xml.stream.XMLStreamReader</code> interface for the attribute and namespace declarations in the XML document</p> <p>Standard specifications</p> <p>Javadoc for <code>XMLStreamReader</code> contains description related to attribute and namespace parsing, but does not clearly define whether parsing-based events will occur.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>ATTRIBUTE</code> event type and <code>NAMESPACE</code> event type are not returned.</p> <p>Process the attribute and namespace when the <code>START_ELEMENT</code> event type is returned.</p>
6	<p>Condition</p> <p>When you specify a value exceeding the number of attributes or namespace declarations, in the <code>index</code> argument of the following methods, among the methods of the <code>XMLStreamReader</code> interface:</p> <ul style="list-style-type: none"> • <code>getAttributeLocalName(int index)</code> • <code>getAttributeName(int index)</code> • <code>getAttributeNamespace(int index)</code> • <code>getAttributePrefix(int index)</code> • <code>getAttributeType(int index)</code> • <code>getAttributeValue(int index)</code> • <code>getNamespaceURI(int index)</code> • <code>isAttributeSpecified(int index)</code> <p>Standard specifications</p> <p>No operations are defined.</p> <p>Operations of Cosminexus XML Processor</p> <p>The return value for the <code>isAttributeSpecified(int index)</code> method is false.</p> <p>The return value for the other methods is null.</p>
7	<p>Condition</p> <p>When you execute the <code>getTextCharacters(int sourceStart, char[] target, int targetStart, int length)</code> method of the <code>XMLStreamReader</code> interface for the <code>COMMENT</code> event type</p> <p>Standard specifications</p> <p>The descriptions in Javadoc and <code>XMLStreamReader</code> differ as follows:</p> <ul style="list-style-type: none"> • Javadoc Acquire the text related to <code>CHARACTERS</code>, <code>SPACE</code>, or <code>CDATA</code>. • <code>XMLStreamReader</code> The <code>getTextCharacters</code> method is enabled for the <code>COMMENT</code> event type. <p>Operations of Cosminexus XML Processor</p> <p>The text related to the <code>COMMENT</code> event is acquired.</p>
8	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. The <code>getPrefix(String uri)</code> method of the <code>XMLStreamWriter</code> interface is executed. 2. The namespace of the <code>uri</code> argument in 1. is bound to multiple prefixes. <p>Standard specifications</p> <p>No operations are defined.</p> <p>Operations of Cosminexus XML Processor</p> <p>One of the bound prefixes becomes the return value.</p>
9	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. The <code>setPrefix(String prefix, String uri)</code> method of the <code>XMLStreamWriter</code> interface is executed.

No.	Notes
	<p>2. The prefix of the <code>prefix</code> argument in 1. is already bound to a different namespace.</p> <p>Standard specifications</p> <p>No operations are defined.</p> <p>Operations of Cosminexus XML Processor</p> <p>Reset the bind destination prefix for the URI specified in <code>setPrefix</code>.</p>
10	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The XML document is encoded by using UTF-16BE or UTF-16LE. • The encoding specified in the XML declaration is UTF-16. • The encoding for the XML document is obtained by using the <code>getEncoding</code> method of <code>XMLStreamReader</code> or the <code>getCharacterEncodingScheme</code> method of <code>StartDocument</code>. <p>Standard specifications</p> <p>The standard specifications do not define whether to return the encoding for the XML document or for the XML declaration.</p> <p>Operations of Cosminexus XML Processor</p> <p>The encoding for the XML document (UTF-16BE or UTF-16LE) is returned instead of the encoding specified in the XML declaration.</p>
11	<p>Condition</p> <p>When you specify a namespace URI in the arguments of the <code>createStartElement</code> method and the <code>createEndElement</code> method, or the <code>createAttribute</code> method of the <code>XMLEventFactory</code> class</p> <p>Standard specifications</p> <p>The standard specifications do not define whether to write the namespace URI specified in the argument as the namespace declaration in the output XML document.</p> <p>Operations of Cosminexus XML Processor</p> <p>The namespace URI specified in the argument is not written as the namespace declaration in the output XML document. Use the <code>createNamespace</code> method to explicitly create the namespace declaration.</p>
12	<p>Condition</p> <p>When you parse an external entity by using the <code>XMLStreamReader</code> interface</p> <p>Standard specifications</p> <p>The standard specifications do not define whether to deploy the external entity.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>EntityReference</code> event is not generated.</p> <p>An event (such as <code>CHARACTERS</code> or <code>START_ELEMENT</code>) is generated for the entity that deploys the entity reference.</p>
13	<p>You cannot specify "" in the <code>prefix</code> argument of the <code>createAttribute(String prefix, String namespaceURI, String localName, String value)</code> method for the <code>XMLEventFactory</code> class.</p>
14	<p>Do not specify a string other than a space in the <code>content</code> argument of the <code>createIgnorableSpace(String content)</code> method for the <code>XMLEventFactory</code> class.</p>
15	<p>The default value of the <code>javax.xml.stream.isSupportingExternalEntities</code> property in the <code>XMLInputFactory</code> class is <code>true</code>.</p>
16	<p>To specify <code>null</code> in the following properties of the <code>XMLInputFactory</code> class, use the <code>setEventAllocator</code> method, the <code>setXMLReporter</code> method, or the <code>setXMLResolver</code> method. Do not use the <code>setProperty</code> method.</p> <ul style="list-style-type: none"> • <code>javax.xml.stream.allocator</code> • <code>javax.xml.stream.reporter</code> • <code>javax.xml.stream.resolver</code>
17	<p>The specification of the following properties of the <code>XMLInputFactory</code> class is not affected even if you specify <code>false</code> as the value of the <code>javax.xml.stream.supportDTD</code> property:</p> <ul style="list-style-type: none"> • <code>javax.xml.stream.isReplacingEntityReferences</code> • <code>javax.xml.stream.isSupportingExternalEntities</code>

No.	Notes
18	<p>Condition</p> <p>If the input XML contains the following elements and attributes in the <code>add(XMLStreamReader reader)</code> method of the <code>XMLStreamWriter</code> class, the format of output XML and input XML is different:</p> <ul style="list-style-type: none"> • Empty element (example: <code><element1/></code>) • The encoding attribute of the XML declaration • The standalone attribute of the XML declaration <p>Operations of Cosminexus XML Processor</p> <ul style="list-style-type: none"> • Only the start and end tags are output for the empty element (example: <code><element1></element1></code>). • <code><?xml version="1.0" ?></code> is output for the XML declaration.
19	The <code>getNamespaceContext()</code> method returns <code>null</code> for the <code>StartElement</code> event generated by using the <code>createStartElement</code> method of the <code>XMLEventFactory</code> class.
20	You cannot use the ISO-10646-UCS-4 encoding.
21	You cannot use XML version 1.1 documents.
22	<p>You cannot specify supplementary characters in the following parts of StAX:</p> <ul style="list-style-type: none"> • Public identifiers in DTD • System identifiers in DTD • Entities to be parsed internally • Attribute value • Comments • CData
23	<p>The <code>Characters</code> event might be reported separately.</p> <p>The StAX parser can also divide a single string data into multiple strings (chunk) and report data to an application as multiple <code>Characters</code> events. Therefore, you need to be aware that the <code>Characters</code> event occurs continuously on the application side.</p>
24	<p>Condition</p> <p>When you execute the <code>getName()</code> method of the <code>XMLStreamReader</code> interface for event types other than <code>START_ELEMENT/END_ELEMENT</code></p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>IllegalArgumentException</code> exception occurs. The <code>IllegalStateException</code> exception does not occur.</p>
25	<p>Condition</p> <p>When you execute the <code>getPrefix()</code> or <code>getAttributePrefix(int index)</code> methods of the <code>XMLStreamReader</code> interface for the elements without prefixes</p> <p>Operations of Cosminexus XML Processor</p> <p>The return value is <code>""</code> (<code>XMLConstants.DEFAULT_NS_PREFIX</code>).</p>
26	<p>Condition</p> <p>When you use the <code>getAttributeValue(String namespaceURI, String localName)</code> method of the <code>XMLStreamReader</code> interface to specify <code>null</code> in the <code>namespaceURI</code> argument and the local name of the attribute in the <code>localName</code> argument for the attributes with namespaces</p> <p>Operations of Cosminexus XML Processor</p> <p>The return value is <code>null</code>. The attribute value is not acquired.</p>
27	<p>Condition</p> <p>When you execute the <code>getEncoding()</code> method or the <code>getVersion()</code> method of the <code>XMLStreamReader</code> interface for the <code>END_DOCUMENT</code> or the <code>ENTITY_REFERENCE</code> event types</p> <p>Operations of Cosminexus XML Processor</p> <p>For the <code>END_DOCUMENT</code> event type, the <code>NullPointerException</code> exception occurs.</p> <p>For the <code>ENTITY_REFERENCE</code> event type, the return value is <code>null</code>.</p>
28	<p>Condition</p> <p>When you execute the <code>isAttributeSpecified(int index)</code> method of the <code>XMLStreamReader</code> interface</p>

No.	Notes
	Operations of Cosminexus XML Processor The return values differ as follows: <ul style="list-style-type: none"> For attributes specified explicitly in the XML document: <code>true</code> For default attributes specified implicitly in DTD: <code>false</code>
29	Condition When you execute the <code>hasName()</code> method of the <code>XMLStreamReader</code> interface for the <code>ENTITY_REFERENCE</code> event type Operations of Cosminexus XML Processor The return value is <code>true</code> .
30	Condition When you execute the <code>getText()</code> method of the <code>XMLStreamReader</code> interface for the <code>DTD</code> event type Operations of Cosminexus XML Processor "String value of the DOCTYPE declaration" is returned. Same is the case when the DTD internal subset exists. "String value of DTD internal subset" is not returned.
31	Condition When you execute the <code>getElementText()</code> method of the <code>XMLStreamReader</code> interface for the contents of an element containing an internal entity reference. Operations of Cosminexus XML Processor The return value is a string with the replacement text of the internal entity reference concatenated twice.
32	Condition When one of the following conditions is satisfied: <ul style="list-style-type: none"> An XML document without an <code>ENTITY</code> declaration is read by using the <code>XMLStreamReader</code> interface and the <code>getProperty("javax.xml.streamnotations")</code> method of the <code>XMLStreamReader</code> interface is executed in the <code>DTD</code> event type. An XML document without a <code>NOTATION</code> declaration is read by using the <code>XMLStreamReader</code> interface and the <code>getProperty("javax.xml.stream.entities")</code> method of the <code>XMLStreamReader</code> interface is executed in the <code>DTD</code> event type. Operations of Cosminexus XML Processor The return value is <code>java.util.List</code> without an element.
33	Condition When all the following conditions are satisfied: <ol style="list-style-type: none"> A <code>DTD</code> declaration is specified in a parameter entity definition. The parameter entity specified in 1. is used in the <code>DOCTYPE</code> declaration. The XML documents from 1. and 2. are parsed by using <code>StAX</code>. The <code>DTD</code> declaration text is acquired by using the <code>getText</code> method of <code>XMLStreamReader</code> or by the <code>javax.xml.stream.events.DTD</code> interface. Operations of Cosminexus XML Processor The acquired <code>DTD</code> declaration text is invalid.
34	Condition When an invalid string is specified in the attribute value of the pseudo attribute in the XML declaration Operations of Cosminexus XML Processor The <code>com.cosminexus.stax.xml.stream.internal.xni.XNIException</code> exception occurs. Also, a detailed string might not be output within <code>{}</code> (curly brackets) in the message. (Example) <pre>com.cosminexus.stax.xml.stream.internal.xni.XNIException: The standalone document declaration value must be "yes" or "no", not "{0}".</pre>
35	Condition When you execute one of the following methods: <ul style="list-style-type: none"> <code>XMLStreamException.getLocation()</code> method

No.	Notes
	<ul style="list-style-type: none"> • <code>XMLEvent.getLocation()</code> method • <code>XMLStreamReader.getLocation()</code> method <p>Operations of Cosminexus XML Processor The <code>Location</code> information is obtained as an approximate value for identifying the location where the exception occurred.</p>
36	<p>Condition When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. The XML document contains an element name beginning with: (colon). 2. The namespace for the StAX parser is enabled. 3. The XML document from 1. is parsed by using the StAX parser from 2. <p>Standard specifications A parsing error occurs.</p> <p>Operations of Cosminexus XML Processor Parsing terminates normally.</p>
37	<p>Condition When you specify <code>true</code> in the <code>standalone</code> parameter of the <code>createStartDocument(String encoding, String version, boolean standalone)</code> method of the <code>XMLEventFactory</code> class</p> <p>Standard specifications An XML document is generated with the <code>standalone</code> status <code>true</code>.</p> <p>Operations of Cosminexus XML Processor The <code>standalone</code> declaration is not generated.</p>
38	<p>Condition When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • An XML document without version declaration is parsed with <code>XMLEventReader</code>. • The <code>getVersion</code> method of the <code>StartDocument</code> interface is executed <p>Standard specifications Returns the default value <code>1.0</code>.</p> <p>Operations of Cosminexus XML Processor Returns <code>null</code>.</p>
39	<p>Condition When you execute the <code>nextEvent()</code> method of the <code>XMLEventReader</code> class for the following parts in the XML document:</p> <ul style="list-style-type: none"> • Blank string • CDATA section • Notation declaration <p>Operations of Cosminexus XML Processor</p> <ul style="list-style-type: none"> • The blank string becomes the <code>CHARACTERS</code> event type, and not the <code>SPACE</code> event type. • The CDATA section becomes the <code>CHARACTERS</code> event type and not the <code>CDATA</code> event type. • The <code>NOTATION_DECLARATION</code> event does not occur for the notation declaration.
40	<p>Condition When you execute the <code>getEventType()</code> method of the <code>XMLStreamReader</code> interface for the following parts in the XML document:</p> <ul style="list-style-type: none"> • Blank string • CDATA section <p>Operations of Cosminexus XML Processor Both blank string and CDATA section become the <code>CHARACTERS</code> event type, therefore, process both as <code>CHARACTERS</code> event types.</p>
41	<p>Condition When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. <code>namespaceURI</code> is not bound to a prefix.

No.	Notes
	<p>2. 1. is specified in the namespaceURI argument of the following method of the XMLStreamWriter interface:</p> <pre>writeAttribute(String prefix, String namespaceURI, String localName, String value)</pre> <p>Operations of Cosminexus XML Processor</p> <p>Only the attribute is output. The namespace declaration (xmlns:prefix="namespaceURI") is not output.</p>
42	<p>You cannot use the writeAsEncodedUnicode(Writer writer) method for the following interfaces included in the javax.xml.stream.events package:</p> <ul style="list-style-type: none"> • Attribute • Characters • Comment • DTD • EndDocument • EndElement • EntityDeclaration • EntityReference • Namespace • NotationDeclaration • ProcessingInstruction • StartDocument • StartElement • XMLEvent
43	<p>You cannot use the allocate(XMLStreamReader reader, XMLEventConsumer consumer) method of the XMLEventAllocator interface included in the javax.xml.stream.util package.</p>
44	<p>In multi-thread programming, the XMLInputFactory class is not thread-safe. Therefore, make sure that multiple threads do not access the same XMLInputFactory instance concurrently.</p> <p>Use either of the following methods to avoid competition between threads:</p> <ul style="list-style-type: none"> • Each thread has one XMLOutputFactory instance. • Each thread accesses the XMLOutputFactory instance exclusively.
45	<p>In multi-thread programming, the XMLOutputFactory class is not thread-safe. Therefore, make sure that multiple threads do not access the same XMLOutputFactory instance concurrently.</p> <p>Use either of the following methods to avoid competition between threads:</p> <ul style="list-style-type: none"> • Each thread has one XMLOutputFactory instance. • Each thread accesses the XMLOutputFactory instance exclusively.
46	<p>In multi-thread programming, the XMLEventFactory class is not thread-safe. Therefore, make sure that multiple threads do not access the same XMLEventFactory instance concurrently.</p> <p>Use one of the following methods to avoid competition between threads:</p> <ul style="list-style-type: none"> • Each thread has one XMLEventFactory instance. • Each thread accesses the XMLEventFactory instance exclusively.
47	<p>In multi-thread programming, the XMLStreamReader interface is not thread-safe. Therefore, make sure that multiple threads do not use the same XMLStreamReader instance concurrently.</p> <p>To avoid competition between threads, each thread must have one XMLStreamReader instance.</p>
48	<p>In multi-thread programming, the XMLStreamWriter interface is not thread-safe. Therefore, make sure that multiple threads do not use the same XMLStreamWriter instance concurrently.</p> <p>To avoid competition between threads, each thread must have one XMLStreamWriter instance.</p>
49	<p>In multi-thread programming, the XMLEventReader class is not thread-safe. Therefore, make sure that multiple threads do not use the same XMLEventReader instance concurrently.</p> <p>To avoid competition between threads, each thread must have one XMLEventReader instance.</p>

No.	Notes
50	<p>In multi-thread programming, the <code>XMLEventWriter</code> class is not thread-safe. Therefore, make sure that multiple threads do not use the same <code>XMLEventWriter</code> instance concurrently.</p> <p>To avoid competition between threads, each thread must have one <code>XMLEventWriter</code> instance.</p>
51	<p>The <code>namespace</code> argument of the <code>resolveEntity</code> method for the <code>javax.xml.stream.XMLResolver</code> interface is always <code>null</code>.</p>
52	<p>Among the return values of the <code>resolveEntity</code> method for the <code>javax.xml.stream.XMLResolver</code> interface, the <code>XMLStreamReader</code> type and the <code>XMLEventReader</code> type objects are not supported.</p> <p>Standard specifications</p> <p>These resources have one of the following return value types:</p> <ul style="list-style-type: none"> • <code>java.io.InputStream</code> • <code>javax.xml.stream.XMLStreamReader</code> • <code>java.xml.stream.XMLEventReader</code>
53	<p>The return value of the <code>getLocation()</code> method for the following events is <code>null</code>:</p> <ul style="list-style-type: none"> • <code>javax.xml.stream.events.Attribute</code> • <code>javax.xml.stream.events.Namespace</code> • <code>javax.xml.stream.events.NotationDeclaration</code> • <code>javax.xml.stream.events.EntityDeclaration</code>
54	<p>Among the <code>XMLOutputFactory</code> classes, you cannot specify <code>StAXResult</code> for which the system identifier is not specified in the <code>result</code> argument of the following methods:</p> <ul style="list-style-type: none"> • <code>createXMLEventWriter(Result result)</code> • <code>createXMLStreamWriter(Result result)</code>
55	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • <code>true</code> is specified for the <code>javax.xml.stream.isRepairingNamespaces</code> property. • One of the following methods is executed for the <code>XMLStreamWriter</code> interface: <ul style="list-style-type: none"> <code>writeAttribute(prefix, namespaceURI, localName, value)</code> <code>writeStartElement(prefix, localName, namespaceURI)</code> <code>writeEmptyElement(prefix, localName, namespaceURI)</code> • The <code>namespaceURI</code> argument is the same as the default namespace URI. • The <code>prefix</code> argument is neither <code>"</code> nor <code>null</code>. <p>Standard specifications</p> <p>The prefix is not written.</p> <p>Operations of Cosminexus XML Processor</p> <p>The prefix is written.</p>
56	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • <code>true</code> is specified for the <code>javax.xml.stream.isRepairingNamespaces</code> property. • One of the following methods is executed for the <code>XMLStreamWriter</code> interface: <ul style="list-style-type: none"> <code>writeAttribute(prefix, namespaceURI, localName, value)</code> <code>writeStartElement(prefix, localName, namespaceURI)</code> <code>writeEmptyElement(prefix, localName, namespaceURI)</code> • The <code>namespaceURI</code> argument is bound, but to a different prefix. <p>Standard specifications</p> <p>The prefix is generated randomly.</p> <p>(Example)</p> <pre>xmlns:{generated}="namespaceURI" {generated}:localName="value" <{generated}:localName xmlns:{generated}="namespaceURI"></pre> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	<p>A random prefix is not generated.</p> <p>(Example)</p> <pre>xmlns:prefix="namespaceURI" prefix:localName="value" <prefix:localName xmlns:prefix="namespaceURI"></pre>
57	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>javax.xml.stream.isRepairingNamespaces</code> property is not specified, or <code>false</code> is specified. • The <code>writeAttribute(prefix, namespaceURI, localName, value)</code> method of the <code>XMLStreamWriter</code> interface is executed. • The <code>namespaceURI</code> argument is not bound. • The <code>prefix</code> argument is neither <code>"</code> nor <code>null</code>. <p>Standard specifications</p> <p>The namespace declaration is generated.</p> <p>Operations of Cosminexus XML Processor</p> <p>The namespace declaration is not generated.</p>
58	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>javax.xml.stream.isRepairingNamespaces</code> property is not specified, or <code>false</code> is specified. • One of the following methods of the <code>XMLStreamWriter</code> interface is executed: <pre>writeAttribute(prefix, namespaceURI, localName, value) writeStartElement(prefix, localName, namespaceURI) writeEmptyElement(prefix, localName, namespaceURI)</pre> • The <code>namespaceURI</code> argument is bound, but to a different prefix. <p>Standard specifications</p> <p>The <code>XMLStreamException</code> exception occurs.</p> <p>Operations of Cosminexus XML Processor</p> <p>There is no error. The value is written as follows:</p> <ul style="list-style-type: none"> • <code>prefix:localName="value"</code> • <code><prefix:localName></code>
59	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>javax.xml.stream.isRepairingNamespaces</code> property is not specified, or <code>false</code> is specified. • One of the following methods of the <code>XMLStreamWriter</code> interface is executed: <pre>writeAttribute(prefix, namespaceURI, localName, value) writeStartElement(prefix, localName, namespaceURI) writeEmptyElement(prefix, localName, namespaceURI)</pre> • The <code>namespaceURI</code> argument is the same as the default namespace URI. • The <code>prefix</code> argument is neither <code>"</code> nor <code>null</code>. <p>Standard specifications</p> <p>The prefix is not written.</p> <p>Operations of Cosminexus XML Processor</p> <p>The following prefixes are written:</p> <ul style="list-style-type: none"> • <code>prefix:localName="value"</code> • <code><prefix:localName></code>
60	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>javax.xml.stream.isRepairingNamespaces</code> property is not specified, or <code>false</code> is specified. • The <code>writeAttribute(prefix, namespaceURI, localName, value)</code> method of the <code>XMLStreamWriter</code> interface is executed.

No.	Notes
	<ul style="list-style-type: none"> • The <code>namespaceURI</code> argument is the same as the default namespace URI. • The <code>prefix</code> argument is "" or null. <p>Standard specifications The prefix is not written.</p> <p>Operations of Cosminexus XML Processor The <code>XMLStreamException</code> exception occurs.</p>
61	<p>Condition When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • <code>true</code> is specified for the <code>javax.xml.stream.isRepairingNamespaces</code> property. • The <code>writeEmptyElement(prefix, localName, namespaceURI)</code> method of the <code>XMLStreamWriter</code> interface is executed. • The <code>prefix</code> argument is "". <p>Standard specifications If <code>prefix == "" null</code>, the value is processed as the default namespace. The prefix is not generated or written. If <code>namespaceURI</code> is not bound, the <code>xmlns</code> declaration is generated or written.</p> <p>Operations of Cosminexus XML Processor The prefix is generated randomly. The value is not processed as the default namespace.</p>
62	<p>Condition When you specify an XML document for which ISO-10646-UCS-2 is specified in the <code>encoding</code> argument and the <code>encoding</code> pseudo attribute ISO-10646-UCS-2 is specified in the <code>stream</code> argument of the following methods of the <code>XMLInputFactory</code> class:</p> <ul style="list-style-type: none"> • <code>createXMLEventReader(stream, encoding)</code> • <code>createXMLStreamReader(stream, encoding)</code> <p>Operations of Cosminexus XML Processor The <code>com.cosminexus.stax.xml.stream.internal.xni.XNIException</code> exception occurs.</p> <p>Workaround Process by using the <code>createXMLEventReader</code> or <code>createXMLStreamReader</code> methods in which <code>encoding</code> is not specified.</p>
63	<p>Condition When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. The <code>getNamespaceContext()</code> method of the <code>StartElement</code> event is used to obtain <code>NamespaceContext</code>. 2. The namespace URI of <code>xml</code> and <code>xmlns</code> prefixes is obtained for <code>NamespaceContext</code> acquired in 1. <p>Standard specifications</p> <ul style="list-style-type: none"> • The prefix <code>xml</code> is bound to <code>http://www.w3.org/XML/1998/namespace</code>. • The prefix <code>xmlns</code> is bound to <code>http://www.w3.org/2000/xmlns/</code>. <p>Operations of Cosminexus XML Processor The namespace context does not have information on <code>xml</code> and <code>xmlns</code> prefix.</p>
64	<p>When an XML document is read with <code>XMLEventReader</code>, irrespective of the presence or absence of the standalone attribute, the return value of the <code>standaloneSet()</code> method of the <code>StartDocument</code> event is <code>true</code>.</p> <p>Standard specifications The return value is <code>true</code> only if the standalone attribute is set up in the encoding declaration of the XML document.</p>
65	<p>The <code>getEntities()</code> and <code>getNotations()</code> methods of the DTD event always return null.</p> <p>Workaround You can access the information by using the <code>javax.xml.stream.entities</code> and <code>javax.xml.stream.notations</code> properties. For details, see Javadoc for <code>XMLStreamReader</code>.</p>
66	<p>Condition</p>

No.	Notes
	<p>When you specify a type for which the system identifier is not specified for the <code>createXMLEventReader (Source source)</code> or <code>createXMLStreamReader (Source source)</code> methods of the <code>XMLInputFactory</code> class. The applicable types are as follows:</p> <ul style="list-style-type: none"> • <code>DOMSource</code> • <code>SAXSource</code> • <code>StAXSource</code> <p>Operations of Cosminexus XML Processor The <code>XMLStreamException</code> exception occurs.</p>
67	The return value of the <code>getProcessedDTD ()</code> method of the <code>DTD</code> event is always <code>null</code> .
68	<p>The return value of the <code>getNamespaces ()</code> method of the <code>EndElement</code> event is always a blank <code>Iterator</code>.</p> <p>Operations of Cosminexus XML Processor An out-of-range namespace <code>Iterator</code> is returned.</p>
69	<p>Condition When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. The character data in an XML document contains an independent end of CDATA section delimiter (<code>]]></code>). <p>(Example) <pre><?xml version="1.0" encoding="UTF-8" ?> <root>te]]>xt</root></pre> </p> <ol style="list-style-type: none"> 2. The character data from 1. is read by using the <code>XMLStreamReader</code> or <code>XMLEventReader</code> interface and the following syntax parsing event is acquired by using the <code>next ()</code> method. <p>Standard specifications The <code>XMLStreamException</code> exception is thrown from the <code>XMLStreamReader</code> interface or <code>XMLEventReader</code> interface.</p> <p>Operations of Cosminexus XML Processor For the <code>XMLStreamReader</code> interface The execution result of the <code>getTextXXXX ()</code> method that can be executed when the event type of the <code>XMLStreamReader</code> interface is <code>CHARACTERS</code>, contains <code>]]></code>. If you execute the <code>getText ()</code> method, a string containing <code>]]></code> is returned. If you execute the <code>getTextCharacters ()</code> method, an array containing <code>]]></code> is returned. If you execute the <code>getTextCharacters (int sourceStart, char[] target, int targetStart, int length)</code> method, <code>]]></code> is included in the char array <code>target</code>. If you execute the <code>getTextLength ()</code> method, the length of the string that contains <code>]]></code> is returned.</p> <p>For the <code>XMLEventReader</code> interface If you execute the <code>getData ()</code> method for the <code>Characters</code> object returned by the <code>nextEvent ()</code> method, a string containing <code>]]></code> is returned.</p>
70	If the output encoding is not specified in the <code>createXMLEventWriter (OutputStream stream)</code> and <code>createXMLStreamWriter (OutputStream stream)</code> methods of the <code>XMLOutputFactory</code> class, <code>XMLEventWriter</code> , and <code>XMLStreamWriter</code> are output by using the default system character code.
71	The execution results of the <code>createStartDocument ()</code> method of the <code>XMLEventFactory</code> class are same as when UTF-8 is specified in the encoding argument of the <code>createStartDocument (String encoding, ...)</code> method.
72	<p>Condition When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The output encoding specified in the <code>createXMLEventWriter</code> and <code>createXMLStreamWriter</code> methods of the <code>XMLOutputFactory</code> class is not UTF-8 or UTF-16. • An XML document containing a 2-byte code is output with the created <code>XMLEventWriter</code> and <code>XMLStreamWriter</code>. • The encoding specified for the XML document to be output satisfies either of the following conditions: The output encoding specified in the <code>createXMLEventWriter</code> method of the <code>XMLOutputFactory</code> class does not match the encoding pseudo attribute of the output XML declaration specified in the <code>createStartDocument</code> method of the <code>XMLEventFactory</code> class.

No.	Notes
	<p>The encoding argument is not specified in the <code>writeStartDocument</code> method of the <code>XMLStreamWriter</code> interface.</p> <p>Operations of Cosminexus XML Processor</p> <p>An invalid XML document with an output encoding other than UTF-8 and UTF-16, and without the encoding pseudo attribute is output.</p> <p>Workaround</p> <p>Specify UTF-8 in the encoding specification of the <code>createXMLEventWriter</code> method and the <code>createXMLStreamWriter</code> method for the <code>XMLOutputFactory</code> class.</p>
73	<p>Condition</p> <p>When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. <code>Boolean.FALSE</code> is set in the <code>javax.xml.stream.isRepairingNamespaces</code> property of <code>XMLOutputFactory</code>. 2. <code>XMLStreamWriter</code> or <code>XMLEventWriter</code> is created from <code>XMLOutputFactory</code> described in step 1. 3. <code>StAXResult</code> is created from <code>XMLStreamWriter</code> or <code>XMLEventWriter</code> described in step 2. <p>Operations of Cosminexus XML Processor</p> <p>When <code>StAXResult</code> created with the above conditions is specified as an argument in either of the following methods, and an XML document including a namespace declaration is processed, an <code>XMLStreamException</code> exception occurs:</p> <ul style="list-style-type: none"> • <code>validate(Source source, Result result)</code> method of the <code>javax.xml.validation.Validator</code> class • <code>transform(Source source, Result result)</code> method[#] of the <code>javax.xml.transform.Transformer</code> class <p>[#] However, this method is applicable only in the case of an identity transformation without using style sheets.</p> <p>Workaround</p> <p>Specify <code>Boolean.TRUE</code> in the <code>javax.xml.stream.isRepairingNamespaces</code> property of <code>XMLOutputFactory</code>.</p>

6.5 Notes on the Schema Validation

The following table gives cautionary notes on the schema validation.

Table 6–9: Notes on the schema validation

No.	Notes
1	To register an entity resolver with the XML parser, you must always set a system identifier for the return value, <code>InputSource</code> , of the <code>resolveEntity</code> method of the entity resolver. If you do not set the system identifier, resolution of the location of the schema document specified by <code>xsd:import</code> or <code>xsd:include</code> might fail, resulting in an error.

6.6 General Notes on XSLT and XSLTC

The following table gives general cautionary notes on XSLT and XSLTC.

Table 6–10: General notes on XSLT and XSLTC

No.	Notes
1	In multi-thread programming, the <code>TransformerFactory</code> and <code>TransformerFactoryXSLTC</code> classes are not thread-safe. Therefore, multiple threads must not access the same <code>TransformerFactory</code> instance and <code>TransformerFactoryXSLTC</code> instance at the same time. To avoid conflicts between threads, use one of the following methods: <ul style="list-style-type: none">• Each thread has one <code>TransformerFactory</code> instance and one <code>TransformerFactoryXSLTC</code> instance.• Each thread exclusively accesses the <code>TransformerFactory</code> and <code>TransformerFactoryXSLTC</code> instances.
2	In multi-thread programming, the <code>Transformer</code> class is not thread-safe. Therefore, multiple threads must not access the same <code>Transformer</code> instance at the same time. To avoid conflicts between threads, use the following method: <ul style="list-style-type: none">• Each thread has one <code>Transformer</code> instance.
3	In multi-thread programming, the <code>SAXTransformerFactory</code> class, <code>TemplatesHandler</code> and <code>TransformerHandler</code> objects are not thread-safe. Therefore, make sure that each thread exclusively accesses these objects.
4	The XSLT and XSLTC transformers provide the same function, but the messages output when an error occurs might differ.
5	If you specify <code>http://www.w3.org/1999/XSL/Transform</code> for the namespace attribute of the <code>xsl:attribute</code> element, the namespace prefix of the output attribute might differ according to whether XSLT or XSLTC is used. However, the namespace prefix itself has no meaning for a parser with the namespace enabled.
6	When you specify a name other than <code>QName</code> for the name attribute of the <code>xsl:attribute</code> and <code>xsl:processing-instruction</code> element, a warning event occurs in XSLT and an error event occurs in XSLTC.
7	For an error event occurring when there are multiple <code>xsl:decimal-format</code> elements whose attributes except the name attribute have different values, a <code>fatalError</code> event occurs in XSLT, and a warning event occurs in XSLTC.
8	For the name attribute of the <code>xsl:element</code> or <code>xsl:attribute</code> element, do not specify a name with a namespace prefix beginning with <code>xml</code> .
9	When the name attribute of an <code>xsl:element</code> , <code>xsl:attribute</code> , or <code>xsl:processing-instruction</code> element is a variable reference that is an empty character string, a warning event occurs in XSLT and an error event occurs in XSLTC.
10	When there is no name attribute for the <code>xsl:element</code> element, an error event occurs in XSLT and a warning event occurs in XSLTC.
11	If you use the <code>xsl:namespace-alias</code> element, the namespace prefix of the output attribute might differ according to whether XSLT or XSLTC is used. However, the namespace prefix itself has no meaning for a parser with the namespace enabled.
12	For format tokens specified for the format attribute of the <code>xsl:number</code> element, specify either <code>A</code> , <code>a</code> , <code>i</code> , <code>I</code> , <code>1</code> , or an optional number of zeros (0) followed by <code>1</code> .
13	When the level attribute of the <code>xsl:number</code> element is any, the behavior might differ according to whether you use XSLT or XSLTC if you specify a pattern that indicates an attribute node for the count attribute. In XSLT, the attribute node is not counted, but in XSLTC, the attribute node is counted.
14	Do not specify an empty character string for the <code>doctype-system</code> attribute of the <code>xsl:output</code> element. Similarly, never specify an empty character string for the <code>doctype-system</code> property by using the <code>setOutputProperty</code> method of the <code>Transformer</code> class.
15	For the method attribute of the <code>xsl:output</code> element, specify either <code>xml</code> , <code>html</code> , or <code>text</code> .
16	When you type an XPath expression whose evaluation result is an empty character string for the name attribute of the <code>xsl:processing-instruction</code> element, a warning event occurs in XSLT and an error event occurs in XSLTC.
17	If you type a variable reference for the select attribute of the <code>xsl:value-of</code> element, a <code>TransformerException</code> occurs.
18	The evaluation result of <code>document (/)</code> might differ according to whether XSLT or XSLTC is used. In XSLT, <code>document (/)</code> is evaluated as a root element of the stylesheet, but in XSLTC, <code>document (/)</code> is evaluated as an empty set of nodes.

No.	Notes
19	<p>When you specify a character string for the first argument of the document function, an error level might differ according to whether you use XSLTC or XSLT if you specify the following XML document as the character string:</p> <ul style="list-style-type: none"> • XML document that does not exist • XML document that is not well-formed
20	If you include a currency sign (#x00A4) in the format pattern string of the <code>format-number</code> function, an error might not occur.
21	For the argument of the system-property function, specify either <code>xsl:version</code> , <code>xsl:vendor</code> , or <code>xsl:vendor-url</code> .
22	In the transformation using an embedding stylesheet, if there are multiple stylesheet elements corresponding to the ID specified by the <code>pseudo</code> attribute href of the processing instruction <code>xml-stylesheet</code> , the applied template might differ according to whether XSLT or XSLTC is used. In XSLT, the last template to appear is applied, but in XSLTC, the first template to appear is applied.
23	With the transformer function, when an unsupported encoding is specified for the output property, the error message (KECX02322-W) is not reported to the error listener but is output to the standard output stream even if the error listener is registered with the transformer in the application. Note that, in case of XSLTC, the error message (KECX07076-W) is reported to the error listener as well as the above error message being output to the standard output stream. For details on the messages, see <i>11. KECX (Messages Output by Cosminexus XML Processor)</i> in the manual <i>uCosminexus Application Server Messages</i> .
24	When the transformation result is output in HTML format, the linefeed output between elements might differ according to whether XSLT or XSLTC is used. However, as the linefeed between elements has no meaning in the HTML document, there is no problem.
25	<p>When all of the following conditions are met, an <code>IllegalArgumentException</code> instead of <code>IllegalStateException</code> occurs if the <code>setNode(Node node)</code> method of the <code>DOMResult</code> object is executed:</p> <ul style="list-style-type: none"> • <code>nextSibling</code> of the <code>DOMResult</code> object is not null. • <code>nextSibling</code> is not the child of the argument node. <p>In this case, in the catch clause that handles exceptions of the <code>setNode</code> method, you must catch not only <code>IllegalStateException</code> but also <code>IllegalArgumentException</code> exceptions.</p>
26	At the locations evaluated as <code>QName</code> in the stylesheet (such as <code>name</code> attribute of the <code>xsl:element</code> element, and first argument of the <code>key</code> function), the process continues execution without any error even when characters that are not allowed to be used as names (such as Unicode supplementary characters, and double-byte numeric characters) according to the XML1.0 standards are used. In such cases, even though <code>xml</code> is specified as the output method in the transformer, and <code>1.0</code> is specified as the output version, the output document is not compliant to XML1.0 standards.
27	If an element name with 477 or more characters is included in the character string that defines an internal entity, the <code>java.lang.IndexOutOfBoundsException</code> exception might occur.
28	Specify the new <code>Document</code> object created by the <code>newDocument</code> method of the <code>DocumentBuilder</code> class ,when setting the <code>Document</code> object in the constructor argument of <code>DOMResult</code> indicated below:
	<ul style="list-style-type: none"> • <code>DOMResult (Node node)</code> • <code>DOMResult (Node node and String systemId)</code>
29	When an XSLT style sheet using the <code>id</code> functionality is applied to an XML document that includes an ID type attribute with a duplicated value, the result of the <code>id</code> functionality might be invalid.
30	<p>The speed of executing the transformation processing for a transformer decreases greatly,if the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The input XML document contains a large amount of entity references. • The output destination of the transformer is <code>DOMResult</code>.
31	<p>When all the following conditions are satisfied, a <code>fatalError</code> event occurs during the invocation of the <code>newTemplates</code> method or the <code>newTransformer</code> method of the <code>TransformerFactory</code> class. Return value of the method becomes null.</p> <ol style="list-style-type: none"> 1. Instead of XSLTC, XSLT is used as the transformer. 2. <code>XMLStreamReader</code> or <code>XMLEventReader</code> is created by invoking the <code>createXMLStreamReader</code> method or the <code>createXMLEventReader</code> method of the <code>XMLInputFactory</code> class without specifying a system identifier in arguments of the methods. 3. <code>XMLStreamReader</code> or <code>XMLEventReader</code> created in step 2 above is specified in the constructor argument of <code>StAXSource</code>. 4. <code>StAXSource</code> created in step 3 above is specified in the argument of the <code>newTemplates</code> method or the <code>newTransformer</code> method of the <code>TransformerFactory</code> class. <p>Take action using any of the following methods:</p>

No.	Notes
	<ul style="list-style-type: none"> • Use <code>XSLTC</code> instead of <code>XSLT</code>. • Specify a system identifier in the argument of the <code>createXMLStreamReader</code> method or the <code>createXMLEventReader</code> method. • Use a source other than <code>StAXSource</code>.
32	<p>When all the following conditions are specified, the <code>encoding</code> pseudo attribute is not output to the XML declaration of output results of the transformer. Furthermore, if the conditions described in step 1 and step 2 overlap, the <code>standalone</code> pseudo attribute is not output.</p> <ol style="list-style-type: none"> 1. <code>StAXResult</code> is specified in the output of the transformer. 2. <code>xml</code> is specified in the output method of the XSLT style sheet or the settings of the output property of the transformer. 3. In <code>StAXResult</code> described in step 1 above, either <code>XMLStreamWriter</code>, or <code>XMLEventWriter</code> for which an output encoding other than UTF-8 is indicated, is specified.
33	<p>When all the following conditions are specified, an XML declaration is not output in the output result of the transformer:</p> <ul style="list-style-type: none"> • <code>StAXResult</code> is specified in the output of the transformer. • <code>xml</code> is specified in the output method of the XSLT style sheet or the settings of the output property of the transformer. • A node matching the template of the XSLT style sheet does not exist in the input XML document.
34	<p>When all the following conditions are satisfied, regardless of the output results of the transformer, <code>xml</code> is assumed as an output method.</p> <ul style="list-style-type: none"> • <code>StAXResult</code> is specified in the output of the transformer. • No output method is specified in the XSLT style sheet or the settings of the output property of the transformer.
35	<p>When <code>StAXResult</code> is used in the output of the transformer, the following properties specified in the XSLT style sheet or the output property of the transformer are disabled:</p> <ul style="list-style-type: none"> • <code>omit-xml-declaration</code> • <code>doctype-system</code> • <code>doctype-public</code> • <code>cdata-section-elements</code>
36	<p>When <code>StAXResult</code> is used in the output of the transformer, an error is not thrown even if an inappropriate value is specified in the following properties that are specified in the XSLT style sheet or the output property of the transformer, and therefore the specification is disabled:</p> <ul style="list-style-type: none"> • <code>version</code> • <code>encoding</code>
37	<p>When all the following conditions are specified, the content of the <code>xsl:text</code> element, in which <code>yes</code> is specified in the <code>disable-output-escaping</code> attribute within the output result of the transformer, is not escaped and is output:</p> <ul style="list-style-type: none"> • <code>StAXResult</code> is specified in the output of the transformer. • <code>yes</code> is specified in the <code>disable-output-escaping</code> attribute of the <code>xsl:text</code> element in the XSLT style sheet. <p>Furthermore, the following processing instruction is added before and after the output character string:</p> <p>Example:</p> <pre><?javax.xml.transform.disable-output-escaping ?>&lt;<?javax.xml.transform.enable-output-escaping ?></pre>
38	<p>When all the following conditions are specified, characters are output before the XML declaration of the output result of the transformer:</p> <ul style="list-style-type: none"> • <code>StAXResult</code> is specified in the output of the transformer. • <code>xml</code> is specified in the output method of the XSLT style sheet or the settings of the output property of the transformer. • Characters are output before the root element (prologue part) of the output tree.

6.7 Notes on XSLT

The following subsections give cautionary notes on XSLT.

6.7.1 Cases Where XSLT Does Not Report Errors

Table 6–11: Notes on XSLT (cases where XSLT does not report errors)

No.	Notes
1	If an expression that combines operands other than the node set with operators is coded in the <code>select</code> attribute of the <code>xsl:copy-of</code> element, the XSLT transformer does not report the error.
2	When the <code>select</code> attribute of the <code>xsl:for-each</code> element contains a character that is not permitted for the XPath expression, for example, ~ or #, the XSLT transformer does not report the error.
3	If you specify a character string that does not match the QName rule for the <code>cdata-section-elements</code> attribute of the <code>xsl:output</code> element, the XSLT transformer does not report the error.
4	If you specify an argument other than character string type for the <code>element-available</code> and <code>function-available</code> functions, the XSLT transformer does not report the error.

6.7.2 Other Notes

Table 6–12: Notes on XSLT (other notes)

No.	Notes
1	If you specify a negative value unavailable for the <code>grouping-size</code> attribute of the <code>xsl:number</code> element, grouping is carried out after transforming the value to a positive value.
2	Specify a character string other than the empty character string for the <code>elements</code> attribute of the <code>xsl:preserve-space</code> and <code>xsl:strip-space</code> elements.
3	If you perform transformation by using XSLT, when you specify * for the <code>elements</code> attribute of the <code>xsl:strip-space</code> or <code>xsl:preserve-space</code> element, elements with a namespace prefix might not be selected for the target to delete the blank spaces or to be stored. If you select an element with a namespace prefix for the target to delete the blank spaces or to be stored, specify the element name with a namespace prefix for the <code>elements</code> attribute.
4	Specify a format pattern that includes one or more characters for the second argument of the <code>format-number</code> function.
5	If you specify <code>html</code> for the <code>method</code> attribute of the output property and an element name for the <code>cdata-section-elements</code> attribute, the <code>cdata-section-elements</code> attribute is not ignored, and the content of the element specified as a CDATA section is output.

6.8 Notes on XSLTC

The following subsections provide notes on XSLTC:

6.8.1 Notes on the Stylesheet Size

Table 6–13: Notes on XSLTC (stylesheet size)

No.	Notes
1	XSLTC imposes an upper limit on the size of the stylesheets for processing. When processing a stylesheet that exceeds 18,000 bytes, the KECX07058-E error may occur and the processing may be aborted. However, the upper limit on the size of the stylesheet depends on the content of the stylesheet. For details on the messages, see <i>11. KECX (Messages Output by Cosminexus XML Processor)</i> in the manual <i>uCosminexus Application Server Messages</i> .

6.8.2 Notes on Transformation Performance

Table 6–14: Notes on XSLTC (transformation performance)

No.	Notes
1	In all cases described in <i>3.2.1 (1) Transforming multiple XML documents by using the same XSLT stylesheet</i> , it is not guaranteed that the transformation performance of the XSLTC transformer may be higher than that of the XSLT transformer. When using the XSLTC transformer, Hitachi recommends that you evaluate the performance with the XSLT stylesheet that is actually used and target XML document for the transformation.
2	<p>If XSLTC is used for transformation of an XML document that uses a stylesheet that meets all of the following conditions, the memory usage increases in proportion to the number of repetitions of processing. Estimate the memory requirements under the assumption that processing is repeated the maximum possible number of times.</p> <ol style="list-style-type: none">1. Either of the following items is set as a variable or parameter in the <code><xsl:variable></code> or <code><xsl:param></code> element:<ul style="list-style-type: none">- Result tree fragment (value defined by using a template specified as element content rather than the <code>select</code> attribute)- Node set2. The item in condition 1 is processed repeatedly. <p>Example 1: In this example, a result tree fragment is set as a variable:</p> <pre><xsl:for-each select="/ns1:root/ns1:Data"> <xsl:variable name="V1"> <xsl:element name="RTF_ROOT" namespace=""> <xsl:copy-of select="." /> </xsl:element> </xsl:variable> <xsl:element name="ELE1" namespace=""> <xsl:copy-of select="\$V1" /> </xsl:element> </xsl:for-each></pre> <p>Example 2: In this example, a result tree fragment is set as a parameter:</p> <pre><xsl:template match="/"> <xsl:for-each select="/ns1:root/ns1:Data"> <xsl:call-template name="t2"> <xsl:with-param name="P1"> <xsl:element name="RTF1" namespace=""> <xsl:copy-of select="." /> </xsl:element> </xsl:with-param> </xsl:call-template> </xsl:for-each></pre>

No.	Notes
	<pre> </xsl:with-param> </xsl:call-template> </xsl:for-each> </xsl:template> <xsl:template name="t2"> <xsl:param name="P1"/> <xsl:element name="ELE1" namespace=""> <xsl:copy-of select="\$V1" /> </xsl:element> </xsl:template> </pre> <p>Example 3: In this example, a node set is set as a variable (the <code>document()</code> function is used to load different XML documents):</p> <pre> <xsl:for-each select="/ns1:root/ns1:Data"> <xsl:variable name="V1" select="document(@refdoc)"/> <xsl:element name="ELE1" namespace=""> <xsl:copy-of select="\$V1/*" /> </xsl:element> </xsl:for-each> </pre>

6.8.3 Notes on XSLT Elements

Table 6–15: Notes on XSLTC (XSLT elements)

No.	Notes
1	The <code>xsl:apply-imports</code> element is not processed.
2	When the template receives a node set as the parameter by <code>xsl:with-param</code> and you refer to the parameter in the <code>select</code> attribute of the <code>xsl:apply-templates</code> element, you can no longer retrieve the node set even if you refer to the same parameter.
3	If the child element of the <code>xsl:attribute</code> element is <code>xsl:copy</code> , the output attribute value is null.
4	If you specify the <code>xsl:value-of</code> element for the content of the <code>xsl:attribute</code> element, write the XPath expression in the <code>select</code> attribute as an absolute location path.
5	If the <code>xsl:attribute-set</code> element refers to itself (circular reference) in the <code>use-attribute-sets</code> attribute, a stack overflow may occur in XSLTC.
6	If you define multiple <code>xsl:decimal-format</code> elements and every attribute has the same value among those elements, XSLTC reports the error.
7	Do not specify an attribute value template for the <code>name</code> attribute of the <code>xsl:element</code> or <code>xsl:attribute</code> element.
8	<p>When all of the following conditions are met, the namespace to which the element belongs is invalid:</p> <ul style="list-style-type: none"> • There is an <code>xsl:attribute</code> element as a child element of the <code>xsl:element</code> element. • The <code>name</code> attribute of the <code>xsl:element</code> element is named with a prefix, and the <code>namespace</code> attribute is specified. • The <code>name</code> attribute of the <code>xsl:attribute</code> element is named with a prefix, and the <code>namespace</code> attribute is specified. • The prefixes specified in the <code>name</code> attributes of <code>xsl:element</code> and <code>xsl:attribute</code> are identical, and their <code>namespace</code> attribute values are different from each other. <p>An example of the stylesheet that meets the above cases is as follows:</p> <pre> <xsl:template match="/"> <xsl:element name="child1" namespace="AAA"> <xsl:element name="A:child2" namespace="XXX"> <xsl:attribute name="A:attr" namespace="AAA">1</xsl:attribute> </xsl:element> </pre>

No.	Notes
	<pre></xsl:element> </xsl:template></pre>
9	<p>If you specify an XPath expression with a predicate for the <code>select</code> attribute of the <code>xsl:for-each</code> element, do not specify a node-comparison expression for the predicate. In this case, modify the stylesheet as follows:</p> <ol style="list-style-type: none"> 1. Delete the predicate from the XPath expression in the <code>select</code> attribute of the <code>xsl:for-each</code> element. 2. Insert the <code>xsl:if</code> element as a child of the <code>xsl:for-each</code> element, and specify the content of the predicate deleted in step 1 for the <code>test</code> attribute. <p>An example of the modification of the stylesheet is as follows:</p> <p>(Before modification)</p> <pre><xsl:template match="/"> <xsl:for-each select="element[sub='x']"> <xsl:value-of select="."/> </xsl:for-each> </xsl:template></pre> <p>(After modification)</p> <pre><xsl:template match="/"> <xsl:for-each select="element"> <xsl:if test="sub='x'"> <xsl:value-of select="."/> </xsl:if> </xsl:for-each> </xsl:template></pre>
10	<p>For the <code>select</code> attribute of the <code>xsl:for-each</code> element, do not specify an XPath expression that includes <code>descendant::node()</code> or <code>descendant-or-self::node()</code>. If you want to use the XPath expression above in order to select element and text nodes, modify the stylesheet as follows:</p> <ul style="list-style-type: none"> • Replace <code>descendant::node()</code> with <code>descendant::node() descendant::text()</code>. • Replace <code>descendant-or-self::node()</code> with <code>descendant-or-self::node() descendant-or-self::text()</code>.
11	<p>For the <code>test</code> attribute of the <code>xsl:if</code> or <code>xsl:when</code> element, do not specify an XPath expression that has both a self axis and a predicate. In this case, modify the stylesheet as follows:</p> <ol style="list-style-type: none"> 1. Delete the predicate from the XPath expression in the <code>test</code> attribute of the <code>xsl:if</code> or <code>xsl:when</code> element. 2. Insert the <code>xsl:if</code> element as a child of the <code>xsl:if</code> element or <code>xsl:when</code> element, and specify the content of the predicate deleted in step 1 for the <code>test</code> attribute. <p>An example of the modification of the stylesheet is as follows:</p> <p>(Before modification)</p> <pre><xsl:template match="/element/sub"> <xsl:if test="self::sub[attribute::attr='x']"> <xsl:value-of select="."/> </xsl:if> </xsl:template></pre> <p>(After modification)</p> <pre><xsl:template match="/element/sub"> <xsl:if test="self::sub"> <xsl:if test="attribute::attr='x'"> <xsl:value-of select="."/> </xsl:if> </xsl:if> </xsl:template></pre>
12	<p>If the <code>href</code> attribute value of the <code>xsl:import</code> element is an empty character string or only blank space, an error event occurs and null is output as the error message.</p>

No.	Notes
13	Do not specify a pattern that includes <code>node()</code> in the <code>count</code> attribute of the <code>xsl:number</code> element. If you want to use <code>node()</code> in order to number element nodes, use <code>*</code> (asterisk) or the element name instead.
14	If you specify multiple whitespace (tab, space, empty character string) strings, a string of one or more characters, or double-byte characters for the <code>grouping-separator</code> attribute of the <code>xsl:number</code> or the <code>xsl:decimal-format</code> element, an error does not occur and the values delimited by the specified string are output.
15	If you specify <code>/</code> (forward slash) for the <code>count</code> attribute of the <code>xsl:number</code> element, 1 is output in XSLT and a <code>java.lang.VerifyError</code> occurs in XSLTC.
16	If you specify <code>/</code> (forward slash) for the <code>from</code> attribute of the <code>xsl:number</code> element, 1 is output in XSLT and a <code>java.lang.VerifyError</code> occurs in XSLTC.
17	Specify a character string other than the empty character string for the <code>doctype-public</code> attribute of the <code>xsl:output</code> element. Similarly, if you set the <code>doctype-public</code> property by using the <code>setOutputProperty</code> method of the Transformer class, specify a character string other than an empty character string.
18	If you specify XML that is unacceptable as a target name of a processing instruction for the <code>name</code> attribute of the <code>xsl:processing-instruction</code> element, an invalid processing instruction <code><?XML?></code> is output.
19	If you specify an invalid value NUMBER (uppercase) for the <code>data-type</code> attribute of <code>xsl:sort</code> , the default behavior (to sort in the same order as when <code>text</code> is specified) is not performed.
20	If you specify an invalid value DESCENDING (uppercase) for the <code>order</code> attribute of <code>xsl:sort</code> , the default behavior (to sort in the same order as when <code>ascending</code> is specified) is not performed.
21	For the <code>match</code> attribute of the <code>xsl:template</code> element, do not specify an XPath expression that includes <code>node()</code> and is followed by a location path. If you want to use <code>node()</code> in order to select element nodes, use an <code>*</code> (asterisk) instead. An example of the modification of the stylesheet is as follows: (Before modification) <pre><xsl:template match="/node()/sub"> <result><xsl:value-of select="."/></result> </xsl:template></pre> (After modification) <pre><xsl:template match="*/sub"> <result><xsl:value-of select="."/></result> </xsl:template></pre>
22	If you code the top-level element for the child element of the <code>xsl:template</code> element, the top-level element is not ignored and processed as a valid child element.
23	If you code a <code>string</code> function that has a negative zero as the argument for the <code>select</code> attribute of the <code>xsl:value-of</code> element, 0 is output in XSLT, but <code>-0</code> is output in XSLTC.
24	If you specify the <code>xsl:call-template</code> element for the child element of the <code>xsl:variable</code> element and there is a literal element with a namespace prefix in the template that is called by <code>xsl:call-template</code> , the namespace declaration of the element is not copied properly when you try to copy the element indicated by the variable in the <code>xsl:copy-of</code> element.
25	If you set an extended element by specifying the <code>xsl:extension-element-prefixes</code> attribute for the literal result element, the <code>xsl:fallback</code> element coded as a child element of the extended element may be ignored. Instead, specify the <code>extension-element-prefixes</code> attribute for the <code>xsl:stylesheet</code> element.
26	If you code a child element in an unacceptable location of the stylesheet according to the specifications (for example, when a literal element is coded as a child element of the <code>xsl:stylesheet</code> element or when an <code>xsl:text</code> element appears as a child element of the <code>xsl:namespace-alias</code> element), the invalid child element is ignored without an error occurring.

6.8.4 Notes on the XPath Expression

Table 6–16: Notes on XSLTC (XPath expression)

No.	Notes
1	If you specify a document function followed by an XPath expression, use <code>/descendant-or-self::node()</code> instead of <code>//</code> .
2	Do not use the current function in the top-level element.
3	If you use the node test <code>node()</code> in an XPath expression, nodes cannot be obtained properly.
4	If you code an XPath expression that combines a node test including an <code>*</code> (asterisk) with a logical operator as the XPath expression, an error occurs during the parsing of the XPath expression.
5	If the value of a variable or parameter is a result tree fragment (not a value in the select attribute, but a value defined by the template that is coded as an element content), a <code>NullPointerException</code> may occur when there is a variable reference in the template.

6.8.5 Cases where XSLTC Does Not Report Errors

In some cases, for example, when the content of an element or attribute is invalid or when the function argument is invalid, XSLTC may not report errors. The following table lists and describes the cases where XSLTC does not report errors.

Table 6–17: Notes on XSLTC (cases where XSLTC does not report errors)

No.	Notes
1	A stylesheet element other than the <code>xsl:sort</code> and <code>xsl:with-param</code> elements is specified as a child element of the <code>xsl:apply-templates</code> element.
2	An empty character string is specified for the <code>select</code> attribute of the <code>xsl:apply-templates</code> element.
3	Element content other than text is specified for the <code>xsl:comment</code> element.
4	A value that consists of two or more characters is specified for the following attributes of the <code>xsl:decimal-format</code> element: <ul style="list-style-type: none">• <code>decimal-separator</code>• <code>grouping-separator</code>• <code>minus-sign</code>• <code>percent</code>• <code>per-mille</code>• <code>zero-digit</code>• <code>pattern-separator</code>
5	An element name qualified with a prefix is specified for the <code>name</code> attribute of the <code>xsl:element</code> or <code>xsl:attribute</code> element, and an empty character string is specified for the <code>namespace</code> attribute.
6	<code>xsl:sort</code> is specified as a child element for an element other than the first element of the <code>xsl:for-each</code> element.
7	Element content is specified for the <code>xsl:import</code> element.
8	The <code>xsl:import</code> element is not the first element of the top-level element, or the <code>xsl:import</code> or <code>xsl:include</code> element is not the top-level element.
9	An XPath expression that includes the <code>key</code> function is specified for the <code>use</code> attribute of the <code>xsl:key</code> element.
10	An <code>xsl:message</code> element is specified as a child element of the <code>xsl:stylesheet</code> element.
11	A value other than <code>yes</code> or <code>no</code> is specified for the <code>terminate</code> attribute of the <code>xsl:message</code> element.
12	There is no namespace declaration corresponding to the prefix specified for the <code>stylesheet-prefix</code> or <code>result-prefix</code> attribute of the <code>xsl:namespace-alias</code> element.

No.	Notes
13	The <code>stylesheet-prefix</code> or <code>result-prefix</code> attribute of the <code>xsl:namespace-alias</code> element is not specified.
14	Element content is specified for the <code>xsl:namespace-alias</code> element.
15	A character string that does not match the Number rule for the <code>grouping-size</code> attribute of the <code>xsl:number</code> element is specified.
16	An invalid value is specified for the <code>lang</code> and <code>level</code> attributes of the <code>xsl:number</code> element.
17	An invalid value is specified for the <code>letter-value</code> attribute of the <code>xsl:number</code> element.
18	A variable reference whose value is an empty character string is specified for the <code>grouping-separator</code> attribute of the <code>xsl:number</code> element.
19	An invalid value is specified for the <code>version</code> attribute of the <code>xsl:output</code> element.
20	An invalid value is specified for the <code>indent</code> attribute of the <code>xsl:output</code> element.
21	An invalid value is specified for the <code>standalone</code> attribute of the <code>xsl:output</code> element.
22	An invalid value is specified for the <code>omit-xml-declaration</code> attribute of the <code>xsl:output</code> element.
23	An empty character string is specified for the <code>select</code> attribute of the <code>xsl:param</code> element.
24	An invalid value is specified for the <code>data-type</code> , <code>order</code> , or <code>case-order</code> attribute of the <code>xsl:sort</code> element.
25	The <code>data-type</code> , <code>order</code> , or <code>case-order</code> attribute of the <code>xsl:sort</code> element is a variable reference whose value is an empty character string.
26	A value specified in uppercase for the <code>data-type</code> , <code>order</code> , or <code>case-order</code> attribute of the <code>xsl:sort</code> element is handled as specified in lowercase.
27	Element content is specified for the <code>xsl:strip-space</code> , <code>xsl:preserve-space</code> , <code>xsl:apply-imports</code> element, and so on which cannot have any element content according to the specifications.
28	An invalid value is specified for the <code>version</code> attribute of the <code>xsl:stylesheet</code> element.
29	The namespace URI corresponding to the prefix specified for the <code>exclude-result-prefixes</code> attribute of the <code>xsl:stylesheet</code> element is not valid for the elements that have this attribute.
30	The <code>xsl:stylesheet</code> element is specified in the <code>xsl:stylesheet</code> element.
31	An empty character string is specified in the <code>match</code> attribute and in the <code>mode</code> attribute of the <code>xsl:template</code> element.
32	Neither the <code>name</code> attribute nor the <code>match</code> attribute is specified for the <code>xsl:template</code> element.
33	An empty character string is specified for the <code>mode</code> attribute of the <code>xsl:template</code> or <code>xsl:apply-templates</code> element.
34	An empty character string is specified for the <code>priority</code> attribute of the <code>xsl:template</code> element.
35	A literal result element appears before the first <code>xsl:template</code> element.
36	The <code>xsl:template</code> element has a <code>mode</code> attribute but does not have a <code>match</code> attribute.
37	Multiple <code>xsl:with-param</code> elements have the <code>name</code> attribute with the same value.
38	A value other than <code>yes</code> or <code>no</code> is specified for the <code>disable-output-escaping</code> attribute of the <code>xsl:text</code> element.
39	A value other than <code>yes</code> or <code>no</code> is specified for the <code>disable-output-escaping</code> attribute of the <code>xsl:value-of</code> element.
40	An empty character string is specified for the <code>select</code> attribute of the <code>xsl:variable</code> element.
41	Both the <code>select</code> attribute and the text content exist in the <code>xsl:variable</code> or <code>xsl:param</code> element.
42	An empty character string is specified for the <code>select</code> attribute of the <code>xsl:with-param</code> element.
43	The number of arguments of a following function call is invalid:

No.	Notes
	<ul style="list-style-type: none"> • number function • boolean function • lang function • concat function
44	The second argument of the document function is an empty character string.
45	An empty character string is specified for the third argument of the format-number function.
46	If a key name that does not exist is specified for the first argument of the key function.
47	If the number of arguments of the unparsed-entity-uri function is invalid.

6.8.6 Other Notes

Table 6–18: Notes on XSLTC (other notes)

No.	Notes
1	If you use XSLTC, the default error listener throws an exception in case of an error.
2	When the implementation of the <code>warning</code> or <code>error</code> method on the <code>ErrorListener</code> does not throw an exception and a warning and an error occurs while parsing the stylesheet, the transformation cannot be carried out because a translet cannot be created.
3	The file name and line number of the file that caused the error may be output before the error ID in the error message.
4	When the <code>setContentHandler</code> method of the <code>XMLReader</code> class is used to register (into the parser) <code>TemplatesHandler</code> object created by the <code>newTemplatesHandler</code> method of the <code>SAXTransformerFactory</code> class and to perform transformation, an error is not output even if a stylesheet with an error is processed.
5	If you perform transformation by using <code>SAXTransformerFactory</code> , the error listener is not notified of the stylesheet error, regardless of the error listener setting.
6	If you perform transformation by using XSLTC, a <code>NullPointerException</code> occurs if <code>#default</code> is specified for both the <code>stylesheet-prefix</code> and <code>result-prefix</code> attributes of the same <code>xsl:namespace-alias</code> element.
7	If you perform transformation by using XSLTC, when two templates with different import priority have <code>name</code> and <code>match</code> attributes and also the <code>name</code> attributes have the same value, the template that has a lower import priority is ignored.
8	If you perform transformation by using XSLTC, when the <code>cdata-section-elements</code> attribute of the output properties are set in both the stylesheet and the JAXP interface, the value defined in the stylesheet is overwritten by the value set in the JAXP interface.
9	If you perform transformation by using XSLTC, when you use <code>URIResolver</code> , which is implemented to return <code>DOMSource</code> created from <code>DocumentFragment</code> , the correct transformation is not performed.
10	If you specify the HTML output for the output method, when <code>{name() }</code> or <code>{name(.) }</code> is specified for the <code>name</code> attribute of the <code>xsl:element</code> element, the <code>NullPointerException</code> occurs.
11	<p>If the number of data items in the implementation of XML Processor reaches 65,536 during transformation by using XSLTC, a <code>TransformerException</code> occurs.</p> <p><i>Data item in the implementation</i> means data items created when the following processing is performed:</p> <ul style="list-style-type: none"> • An input XML document is loaded. One data item is created for every 32,768 nodes (that are equivalent to DOM nodes) of an input XML document. • Transformation is performed for an XML document that uses a stylesheet that includes the following items: <code><xsl:variable></code> or <code><xsl:param></code> element in which a node set or result tree fragment (value defined by using a template specified as element content rather than the <code>select</code> attribute) is specified as a variable or parameter (One data item is created each time this element is processed.)

6.9 Notes on the javax.xml.datatype Package

The following table gives cautionary notes on the `javax.xml.datatype` package.

Table 6–19: Notes on the `javax.xml.datatype` package

No.	Notes
1	In multi-thread programming, the objects defined by the <code>javax.xml.datatype</code> package are not thread-safe. Therefore, make sure that each thread exclusively accesses these objects.
2	If the <code>IllegalArgumentException</code> occurs in each method of the <code>DatatypeFactory</code> class, the parameter specified for each method is returned to the <code>getMessage()</code> method.
3	If you set the month field to either 4, 6, 9, or 11 in the <code>newXMLGregorianCalendar</code> , <code>newXMLGregorianCalendarDate</code> , and <code>newXMLGregorianCalendarTime</code> methods of the <code>DatatypeFactory</code> class, an error does not occur even when you set 31 in the day field.
4	Specify the value from 0 to 999 for the millisecond argument of the <code>newXMLGregorianCalendar</code> and <code>newXMLGregorianCalendarTime</code> methods of the <code>DatatypeFactory</code> class.
5	If you compare a <code>Duration</code> object whose month field has the value 1 and a <code>Duration</code> object whose day field has the value 28, the result is as follows: <ul style="list-style-type: none">• If you compare by using the <code>compare</code> method, <code>DatatypeConstants.EQUAL</code> is returned.• If you compare by using the <code>equals</code> method, <code>true</code> is returned.
6	The <code>Duration</code> object created in the <code>newDuration(long durationInMilliseconds)</code> , <code>newDurationDayTime(long durationInMilliseconds)</code> , or <code>newDurationYearMonth(long durationInMilliseconds)</code> method of the <code>DatatypeFactory</code> class is the <code>DatatypeConstants.DURATION_DAYTIME</code> type. Therefore, the month field and day field of this <code>Duration</code> object are set to 0.
7	Specify the value from 0 to 23 for the hour argument of the <code>setHour</code> and <code>setTime</code> methods of the <code>XMLGregorianCalendar</code> class. Also, specify a value from 0 to 23 for the time field of the argument of the <code>newXMLGregorianCalendar</code> and <code>newXMLGregorianCalendarTime</code> methods of the <code>DatatypeFactory</code> class.
8	In the JAXP1.4 specifications, the specifications of the <code>isValid</code> method of the <code>XMLGregorianCalendar</code> class are defined as the <code>Validate</code> instance by <code>getXMLSchemaType()</code> constraints. In Cosminexus XML Processor, <code>true</code> is returned when all the following constraints are met: <ul style="list-style-type: none">• If the month field is 2, the day field is 28 or smaller (other than leap year) or 29 or smaller (leap year).• The year field is not 0.
9	If the argument of the <code>equals</code> method of the <code>XMLGregorianCalendar</code> class is null, a <code>NullPointerException</code> does not occur, but <code>false</code> is returned.
10	The text expression obtained by applying the <code>toString</code> and <code>toXMLFormat</code> methods to the <code>XMLGregorianCalendar</code> object of the <code>DatatypeConstants.GMONTH</code> type is not the character string <code>--MM</code> defined in section 3.2.14 of the specification <i>W3C XML Schema 1.0 Part 2</i> (second edition), but the string <code>--MM--</code> defined in the specifications prior to the second edition.

6.10 Notes on the javax.xml.validation Package

The following table gives cautionary notes on the `javax.xml.validation` package.

Table 6–20: Notes on the `javax.xml.validation` package

No.	Notes
1	In multi-thread programming, the <code>SchemaFactory</code> , <code>Validator</code> , <code>ValidatorHandler</code> , and <code>TypeInfoProvider</code> classes are not thread-safe. Therefore, make sure that each thread exclusively accesses these objects.
2	The URI " <code>http://www.w3.org/2001/XMLSchema</code> " is the only URI that you can specify as the argument of the <code>newInstance(String schemaLanguage)</code> method of the <code>SchemaFactory</code> class.
3	for the <code>isSchemaLanguageSupported</code> method of the <code>SchemaFactory</code> class. Specify <code>http://www.w3.org/2001/XMLSchema</code> . If you specify other schema languages, this method returns <code>false</code> .
4	Specify a schema document with a different <code>targetNamespace</code> for each array element of the <code>schemas</code> argument for the <code>newSchema(Source[] schemas)</code> method of the <code>SchemaFactory</code> class. If you specify multiple schema documents with the same <code>targetNamespace</code> or with no <code>targetNamespace</code> , a schema document for an array element with a lower index is prioritized.
5	If you execute the <code>newSchema(Source[] schemas)</code> method of the <code>SchemaFactory</code> class under the following condition, exceptions may not occur: <ul style="list-style-type: none">• Array elements for the <code>schemas</code> argument may include both valid schema documents and invalid ones. To avoid this problem, use the following workaround: <ol style="list-style-type: none">1. Save exception objects passed to the <code>error</code> and <code>fatalError</code> methods by the error handler for the <code>SchemaFactory</code> object.2. After calling the <code>newSchema</code> method, check for whether an exception object has been saved in step 1, and if it is successfully saved, throw the exception.
6	If you use a schema document including the definition for ID type attributes to validate XML documents by using the <code>validate</code> method of the <code>Validator</code> class, use either the <code>validate(SAXSource)</code> , <code>validate(DOMSource)</code> , or <code>validate(SAXSource, SAXResult)</code> method rather than the <code>validate(DOMSource, DOMResult)</code> method.
7	When all of the following conditions are met, if you execute the <code>validate(Source source, Result result)</code> method of the <code>Validator</code> class, a <code>DOMException</code> may occur: <ol style="list-style-type: none">1. The <code>result</code> argument is a <code>DOMResult</code> object.2. The DOM node that includes the result tree set in the <code>DOMResult</code> object is the <code>Document</code> object with child nodes. Therefore, make sure that the <code>Document</code> object described in condition 2 does not have any child node.
8	If a schema document includes the <code>xsd:ENTITY</code> type or <code>xsd:ENTITIES</code> type, schema validation by using the <code>ValidatorHandler</code> class is not supported. In such cases, perform the schema validation by using other methods such as the <code>Validator</code> class.
9	If the validation fails, a <code>TypeInfo</code> object obtained from the element or attribute that is not defined in the schema document becomes the same as the <code>TypeInfo</code> object of the <code>xsd:anyType</code> type.
10	If you specify the <code>DOMSource</code> object for the <code>Source</code> argument of the <code>validate(Source)</code> and <code>validate(Source, Result)</code> of the <code>Validator</code> class, make sure that the <code>Node</code> object owned by the <code>DOMSource</code> object meets all of the following conditions: <ol style="list-style-type: none">1. Generated by the parser with the namespace enabled. For example, the <code>DocumentBuilder</code> generated by the <code>DocumentBuilderFactory</code> applied with <code>setNamespaceAware(true)</code>.2. Generated by the parser without validation. For example, the <code>DocumentBuilder</code> generated by the <code>DocumentBuilderFactory</code> applied with <code>setValidating(false)</code>.
11	If you specify the <code>SAXSource</code> object for the <code>Source</code> argument of the <code>validate(Source)</code> and <code>validate(Source, Result)</code> of the <code>Validator</code> class and the <code>SAXSource</code> object uses the <code>XMLReader</code> object, the <code>XMLReader</code> object must meet all of the following conditions:

No.	Notes
	<p>1. The object is an XMLReader object with the namespace enabled. For example, the XMLReader object taken out of the SAXParser that is generated by the SAXParserFactory applied with <code>setNamespaceAware(true)</code>.</p> <p>2. The object is an XMLReader object without validation. For example, the XMLReader object taken out of the SAXParser that is generated by the SAXParserFactory applied with <code>setValidating(false)</code>.</p>
12	<p>When all of the following conditions come in succession, the exception <code>DOMException</code> with the exception code <code>NO_MODIFICATION_ALLOWED_ERR</code> might occur:</p> <ol style="list-style-type: none"> 1. The <code>DOMSource</code> object is specified in the <code>newSchema(Source)</code> and <code>newSchema(Source[])</code> arguments of the <code>SchemaFactory</code> class, and these methods are executed. 2. The DOM tree wrapping the <code>DOMSource</code> object specified in condition 1. is updated.
13	<p>Using the <code>newSchema</code> method of the <code>javax.xml.validation.SchemaFactory</code> class, if you want to specify a <code>SAXSource</code> with an XMLReader specified, the XMLReader to be specified must satisfy the following conditions:</p> <ul style="list-style-type: none"> • <code>true</code> is specified for the value of the SAX2 feature <code>http://xml.org/sax/features/namespace-prefixes</code>.
14	<p>When you specify <code>StAXSource</code> and <code>StAXResult</code> in an argument of the <code>validate(Source, Result)</code> method of the <code>Validator</code> class, specify the same encoding in <code>StAXSource</code> and <code>StAXResult</code>.</p>
15	<p>Condition When the <code>getElementTypeInfo()</code> method of the <code>javax.xml.validation.TypeInfoProvider</code> class is executed in the <code>endElement</code> event of <code>ContentHandler</code> set in <code>ValidatorHandler</code>.</p> <p>Operation of Cosminexus XML Processor The <code>IllegalStateException</code> exception is thrown. The <code>TypeInfo</code> object is not returned.</p>

6.11 Notes on the javax.xml.xpath Package

The following table gives cautionary notes on the `javax.xml.xpath` package.

Table 6–21: Notes on the `javax.xml.xpath` package

No.	Notes
1	In multi-thread programming, the <code>XPathFactory</code> class is not thread-safe. Therefore, multiple threads must not access the same <code>XPathFactory</code> instance at the same time. To avoid conflicts between threads, use one of the following methods: <ul style="list-style-type: none">• Each thread has one <code>XPathFactory</code> instance.• Each thread exclusively accesses the <code>XPathFactory</code> instance.
2	In multi-thread programming, <code>XPath</code> instances generated by the <code>newInstance</code> method of the <code>XPathFactory</code> class and <code>XPathExpression</code> instances generated by the <code>compile</code> method of the <code>XPath</code> interface are not thread-safe. Therefore, multiple threads must not access the same <code>XPath</code> instance and <code>XPathExpression</code> instance at the same time. To avoid conflicts between threads, use the following method: <ul style="list-style-type: none">• Each thread has one <code>XPath</code> instance and one <code>XPathExpression</code> instance.
3	The URI <code>http://java.sun.com/jaxp/xpath/dom</code> is the only URI that you can specify as the argument of the <code>newInstance(String uri)</code> method of the <code>XPathFactory</code> class.
4	If you evaluate an XPath expression that refers to a context by using the following methods, specify a non-null context for the <code>item</code> argument: <ul style="list-style-type: none">• <code>evaluate(String expression, Object item, QName returnType)</code> method of the <code>XPath</code> interface• <code>evaluate(String expression, Object item)</code> method of the <code>XPath</code> interface• <code>evaluate(Object item, QName returnType)</code> method of the <code>XPathExpression</code> interface• <code>evaluate(Object item)</code> method on the <code>XPathExpression</code> interface
5	Specify the <code>Document</code> , <code>DocumentFragment</code> , <code>Element</code> , <code>Text</code> , <code>Attr</code> , <code>ProcessingInstruction</code> , or <code>Comment</code> object for the <code>item</code> argument described in <i>No. 4</i> in this table.
6	For a return type of the <code>evaluate</code> method of the class that implements the <code>XPathFunction</code> interface and a return type of the <code>resolveVariable</code> method of the class that implements the <code>XPathVariableResolver</code> interface, specify either the <code>java.lang.String</code> , <code>java.lang.Boolean</code> , <code>java.lang.Number</code> , <code>org.w3c.Node</code> , or <code>org.w3c.NodeList</code> type.
7	If an exception occurs in a method for the <code>javax.xml.xpath</code> package, a detailed message may not be obtained because the return value from the <code>getMessage</code> method that was applied to the exception object can be null. In this case, if you obtain the wrapped exception object by applying the <code>getCause</code> method to the exception object and then apply the <code>getMessage</code> method, you may obtain the detailed message.
8	Specify spaces before and after <code>and</code> , <code>or</code> , <code>mod</code> , and <code>div</code> operators. (Example) <code>1000 div 10</code>
9	An XPath expression that meets all of the following conditions is not supported: <ol style="list-style-type: none">1. Specifies the XPath function that returns a node set for each operand in the <code> </code> operator, and creates a union of the node sets.2. Applies a predicate to the union of the node sets created in condition 1.
10	An XPath expression that meets all of the following conditions is not supported: <ol style="list-style-type: none">1. Uses either the <code>local-name</code>, <code>namespace-uri</code>, or <code>name</code> function.2. Specifies the following expression for the argument of the function described in condition 1. above: <code>self::node()/descendant::prefix1:*(#)</code> In this case, you can work around this problem by changing the argument of the function described in condition 1 to the following expression: <code>self::node()/self::node()/descendant::prefix1:*(#)</code> The <code>prefix1</code> indicates any namespace prefix.
11	An XPath expression that applies the <code>following-sibling</code> axis to the attribute and namespace nodes is not supported.
12	When an operand of the <code> </code> operator is not a location set, the XPath expression is not supported.

No.	Notes
13	If there are multiple namespace nodes where each set of a namespace prefix and namespace URI is the same, an XPath expression that applies an axis to their namespace nodes is not supported.
14	When the <code>id</code> functionality is applied to an XML document that includes an ID type attribute with a duplicated value, the result of the <code>id</code> functionality might be invalid.

6.12 Notes on the org.w3c.dom Package

The following table gives cautionary notes on the `org.w3c.dom` package.

Table 6–22: Notes on the `org.w3c.dom` package

No.	Notes
1	<i>Table 6-23</i> lists the parameters that can be set for the <code>DOMConfiguration</code> object. Parameters that can be set do not match the return values from the <code>getParameterNames</code> and <code>canSetParameter</code> methods. Also, if you specify a value that cannot be set, an exception may not occur. If you specify a value that cannot be set, the operation is not guaranteed.
2	The parameter name for the <code>DOMConfiguration</code> object is not case-sensitive.
3	For the DOM Level 3, new methods are added to the interface for the existing DOM Level 2. For example, the <code>isId</code> method is added to the <code>Attr</code> interface. Usually, no application program implements these interfaces. If the application implements the interfaces, the method implementation added for the DOM Level 3 needs to be added to the application.
4	If you define the default value for an attribute in <code>XMLSchema</code> and delete the <code>Attr</code> node that corresponds to that attribute, the attribute node for which the <code>getSpecified</code> method returns <code>false</code> will not be generated.
5	The <code>normalizeDocument</code> method of the <code>Document</code> object is not supported.
6	The <code>adoptNode</code> method of the <code>Document</code> object is not supported. Instead, use a combination of the <code>importNode</code> method of the <code>Document</code> object and the <code>removeChild</code> method of the <code>Node</code> object.
7	The <code>getInputEncoding</code> method of the <code>Document</code> object and the <code>getInputEncoding</code> method of the <code>Entity</code> object are not supported.
8	When all of the following conditions are met, a <code>DOMException</code> does not occur by executing the <code>renameNode (Node n, String namespaceURI, String qualifiedName)</code> method of the <code>Document</code> object: <ul style="list-style-type: none">• The <code>n</code> argument is an <code>Attr</code> object that does not support namespace.• The <code>namespaceURI</code> argument is null.• The <code>qualifiedName</code> argument is an invalid qualified name. If you specify the <code>Element</code> or <code>Attr</code> object for the <code>n</code> argument and specify null for the <code>qualifiedName</code> argument, a <code>NullPointerException</code> occurs rather than a <code>DOMException</code> .
9	The <code>getBaseURI</code> method of the <code>Attr</code> , <code>Text</code> , <code>EntityReference</code> , <code>CDATASection</code> , <code>DocumentType</code> , and <code>DocumentFragment</code> objects cannot obtain a URI.
10	Apply the <code>replaceWholeText (String)</code> method of the <code>Text</code> object to the DOM tree without a <code>EntityReference</code> node.
11	A return value from the <code>getByteOffset</code> of the <code>DOMLocator</code> object is always <code>-1</code> .
12	An object of the return value when you apply the <code>getRelatedData</code> method to the <code>DOMError</code> argument of the <code>handleError</code> method of the <code>DOMErrorHandler</code> object is not always a <code>Node</code> object. The return value can be the <code>String</code> object indicating an error message.
13	For a <code>Document</code> object acquired by parsing, the <code>DOMException</code> exception might not occur even when the <code>insertBefore</code> method or the <code>appendChild</code> method is used to add an inappropriate node.
14	When all the following conditions are satisfied, the return value of invoking the <code>getSchemaTypeInfo</code> method for an element node is invalid: <ol style="list-style-type: none">1. Use an XML schema document to define a derived type element using <code>union</code>.2. Use the schema document described in condition 1 above to implement verification check parsing.3. Acquire the element node corresponding to condition 1 above from the DOM that is created in condition 2, and invoke the <code>getSchemaTypeInfo</code> method.

Table 6–23: Parameters that can be set for DOMConfiguration objects

Parameter name	Valid values
canonical-form	false
cdata-sections	true, false
check-character-normalization	false
comments	true, false
datatype-normalization	false
element-content-whitespace	true
entities	true, false
error-handler	DOMErrorHandler object
infoset	true, false
namespaces	true
namespace-declarations	true, false
normalize-characters	false
schema-location	Cannot be set.
schema-type	Cannot be set.
split-cdata-sections	true, false
validate	false
validate-if-schema	false
well-formed	true
resource-resolver	LSResourceResolver object

6.13 Notes on the org.w3c.dom.bootstrap Package

The following table describes the cautionary note on the `org.w3c.dom.bootstrap` package.

Table 6–24: Notes on the `org.w3c.dom.bootstrap` package

No.	Notes
1	Specify non-empty character strings for the argument of the <code>getDOMImplementation</code> method and the argument of the <code>getDOMImplementationList</code> method, which are methods of the <code>DOMImplementationRegistry</code> class.

6.14 Notes on the org.w3c.dom.ls Package

The following table gives cautionary notes on the `org.w3c.dom.ls` package.

Table 6–25: Notes on the `org.w3c.dom.ls` package

No.	Notes
1	You can specify only the <code>DOMImplementationLS.MODE_SYNCHRONOUS</code> for the <code>mode</code> argument of the <code>createLSParser(short mode, String schemaType)</code> method on the <code>DOMImplementationLS</code> interface. Also, you can specify <code>null</code> , <code>http://www.w3.org/2001/XMLSchema</code> , or <code>http://www.w3.org/TR/REC-xml</code> for the <code>schemaType</code> argument.
2	Table 6-26 lists the parameters that can be set for the <code>DOMConfiguration</code> object that is obtained by applying the <code>getDomConfig</code> method to the <code>LSParser</code> object. In addition to those parameters, the parameters listed in Table 6-23 of 6.12 Notes on the org.w3c.dom Package can also be set. Parameters that can be set do not match the return values from the <code>getParameterNames</code> and <code>canSetParameter</code> methods. Also, if you specify a value that cannot be set, an exception may not occur. If you specify a value that cannot be set, the operation is not guaranteed.
3	If you specify any value for the <code>charset-overrides-xml-encoding</code> parameter of the <code>LSParser</code> object, the value is ignored.
4	The <code>parseWithContext</code> method for the <code>LSParser</code> interface is not supported.
5	The <code>setNewLine</code> method of the <code>LSParser</code> object is not supported.
6	The <code>LSParser</code> object cannot analyze the XML1.1 document.
7	The <code>startElement</code> method of the <code>LSParser</code> object is not called for the root element.
8	If a <code>LSEException</code> occurs when you analyze by using the <code>LSParser</code> object or perform serialization by using the <code>LSSerializer</code> object, the return value from the <code>getMessage()</code> method may be <code>null</code> . In such cases, an exception may have occurred in the user implementation class of the <code>LSParserFilter</code> or <code>LSSerializerFilter</code> .
9	Table 6-27 lists the parameters that can be set for the <code>DOMConfiguration</code> object that is obtained by applying the <code>getDomConfig</code> method to the <code>LSSerializer</code> object. In addition to those parameters, the parameters listed in Table 6-23 of 6.12 Notes on the org.w3c.dom Package can also be set. Parameters that can be set do not match the return values from the <code>getParameterNames</code> and <code>canSetParameter</code> methods. Also, if you specify a value that cannot be set, an exception may not occur. If you specify a value that cannot be set, the operation is not guaranteed.
10	If you specify any node other than the <code>Document</code> , <code>DocumentFragment</code> , and <code>Element</code> nodes for the <code>nodeArg</code> argument of the <code>write</code> , <code>writeToString</code> , and <code>writeToURI</code> methods for the <code>LSSerialize</code> object, those nodes are not serialized.
11	The <code>write</code> , <code>writeToString</code> , and <code>writeToURI</code> methods of the <code>LSSerialize</code> object cannot serialize the nodes whose namespaces are disabled.
12	The default value for the end-of-line sequence character obtained from the <code>getNewLine</code> method of the <code>LSSerializer</code> interface is <code>\n</code> .
13	If the processing instruction in the <code>ProcessingInstruction</code> object includes the character string <code>?></code> , no error notification is provided if the <code>ProcessingInstruction</code> object is serialized.
14	When using <code>LSParser</code> to parse, a <code>DOMException</code> exception might not occur even if the contents of the <code>DocumentType</code> object that is acquired from the <code>Document</code> object are changed.

Table 6–26: Parameters that can be set for `DOMConfiguration` objects (`LSParser` objects)

Parameter name	Valid values
<code>charset-overrides-xml-encoding</code>	<code>true</code> , <code>false</code>
<code>disallow-doctype</code>	<code>false</code>
<code>ignore-unknown-character-denormalizations</code>	<code>true</code>

Parameter name	Valid values
supported-media-types-only	false

Table 6–27: Parameters that can be set for DOMConfiguration objects (LSSerializer objects)

Parameter name	Valid values
discard-default-content	true, false
format-pretty-print	false
ignore-unknown-character-denormalizations	true
xml-declaration	true, false

6.15 Notes on the org.xml.sax.ext Package

The following table gives cautionary notes on the `org.xml.sax.ext` package.

Table 6–28: Notes on the `org.xml.sax.ext` package

No.	Notes
1	<p>The <code>addAttribute</code> method of the <code>Attributes2Impl</code> class is not supported.</p> <p>Instead, use the <code>addAttribute</code> method of the <code>AttributesImpl</code> class and modify the application as follows:</p> <p>(Before modification)</p> <pre>Attributes2Impl myAtt = new Attributes2Impl(); String type = "ID"; myAtt.addAttribute("http://hitachi-xmlprocessor.com/", "attr", "ht:attr", type, "value"); ...</pre> <p>(After modification)</p> <pre>AttributesImpl myAtt = new AttributesImpl(); String type = "ID"; myAtt.addAttribute("http://hitachi-xmlprocessor.com/", "attr", "ht:attr", type, "value"); Attributes2Impl myAtt2 = new Attributes2Impl(myAtt); int position = myAtt2.getLength() - 1; myAtt2.setSpecified(position, true); myAtt2.setDeclared (position, !"CDATA".equals(type)); ...</pre>
2	<p>The <code>getXMLVersion</code> and <code>getEncoding</code> methods of the <code>Locator2Impl</code> class are not supported.</p>

6.16 Notes on XInclude

The following table gives cautionary notes on XInclude.

Table 6–29: Notes on XInclude

No.	Notes
1	You can specify only an <code>element()</code> schema and a shorthand pointer as a possible schema for the <code>xpointer</code> attribute of the <code>xi:include</code> element.
2	If you specify a shorthand pointer for the <code>xpointer</code> attribute of the <code>xi:include</code> element, the shorthand pointer must be an attribute value of an attribute that is defined as an ID type in the DTD. You cannot specify an attribute value of an attribute that is defined as <code>xsd:ID</code> type in the XML Schema.

6.17 Notes on implementation-dependent specifications

The following tables describe the specifications that depend on the XML Schema implementation.

Table 6–30: Implementation-dependent specifications for data types

Data type			Maximum value	Maximum number of digits	
duration type (Format: PnYnMnDTnHnMnS)	nY (Year)		X	--	
	nM (Month)		X		
	nD (Day)		X		
	nH (Hour)		X		
	nM (Minute)		X		
	nS (Second)	1 or more seconds	X		
		Less than 1 second	X		
date type (Format: CCYY-MM-DD)	CCYY (If 9999 is exceeded, digits can be added.)		X		
gYearMonth type (Format: CCYY-MM)					
gYear type (Format: CCYY)					
dateTime type (Format: CCYY-MM-DDThh:mm:ss)	CCYY (If 9999 is exceeded, digits can be added.) Less than 1 second (Values smaller than 1 second can be specified to the right of the decimal point after ss.)		X		
time type (Format: hh:mm:ss)	Less than 1 second (Values smaller than 1 second can be specified to the right of the decimal point after ss.)		X		
base64Binary type			Y		Z
hexBinary type					Z
decimal type				Z	
integer type				Z	
nonPositiveInteger type				Z	
nonNegativeInteger type				Z	
negativeInteger type				Z	
positiveInteger type				Z	

Legend:

X: Can be up to 2,147,483,647

Y: Maximum value is not defined

Z: Maximum number of digits depends on the memory

--: Not applicable

Table 6–31: Implementation-dependent specifications for constraining facet

Constraining facet	Maximum value	Maximum number of digits
length	X	--
minLength	X	
maxLength	X	
totalDigits	X	
fractionDigits	X	
maxInclusive	Y	Z
maxExclusive		Z
minInclusive		Z
minExclusive		Z

Legend:

X: Can be up to 2,147,483,647

Y: Maximum value is not defined

Z: Maximum number of digits depends on the memory

--: Not applicable

Table 6–32: Implementation-dependent specifications for the number of occurrences

Number of occurrences	Maximum value	Maximum number of digits
minOccurs	X	--
maxOccurs	X	

Legend:

X: Can be up to 2,147,483,647

--: Not applicable

6.18 Precautions related to the schema cache functionality

The following subsections describe the precautions related to the schema cache functionality.

6.18.1 Precautions related to performance

When you use the disk cache, the overheads increase for the process where the cache is read from the file and de-serialized to the object. Therefore, the performance might decrease than you do not use the cache.

6.18.2 Precautions related to error output

The error occurred during the cache setup is output in the error file and not in the standard error output stream. To confirm the contents output as the error output, execute the `cachestate` command after specifying the `-d` option after the J2EE server is started, and then output the error information.

6.18.3 Precautions related to schema definition files

The following table describes the precautions related to schema definition files:

Table 6–33: Precautions related to schema definition files

No.	Precautions
1	, D that is coded at the end of a specified value of the <code>schema_XXX</code> property is considered as a disk cache specifier, so do not use names ending with , D for schema document files.
2	A schema document that is referenced using <code>xsd:import</code> is always cached in the memory regardless of the specification of schema definition files. The schema document that is referenced using <code>xsd:include</code> or <code>xsd:redefine</code> is merged with grammar objects of the schema document of reference source, so the cache location (memory or a disk) will be the same as the location of the schema document of the reference source.
3	If a schema document that is to be used for verification is specified in the format of <code>InputStream</code> and <code>array of objects</code> within the <code>http://java.sun.com/xml/jaxp/properties/schemaSource</code> property, a cache will not be used during the verification. You can use a cache only when the schema document is specified in any of the following formats: <code>String</code> , <code>File</code> , or <code>InputSource</code> . However, for <code>InputSource</code> , you must use the <code>getSystemId</code> method to acquire a system identifier.
4	When a parser error occurs while setting up a cache, the schema document that caused the error and the schema document referencing that document will not be cached.
5	The internal files (INFO file) used for cache setup and deletion by the schema cache functionality are saved in the <code>cache_info</code> directory (INFO directory) below <code><JAXP_DIR></code> . Therefore, if the disk storage of <code><JAXP_DIR></code> [#] exceeds 100%, the cache functionality might not work normally.
6	The schema cache functionality changes or deletes the directories and files exist in the directory specified for the <code>diskcache_path</code> property. Therefore, do not specify a directory used by the system in the <code>diskcache_path</code> property.
7	When specifying a XML schema document, for which an external reference is to be resolved using the entity resolver, in the schema definition file, resolution of the external reference cannot be performed during the cache setup. Therefore, an attempt to parse the schema document will fail and the cache will not be set up.

#

This is the `jaxp` directory in the directory where Cosminexus is installed.

6.18.4 Precautions related to reset up and deletion of a cache

The following table describes the precautions related to re-set up and deletion of a cache:

Table 6–34: Precautions related to reset up and deletion of a cache

No.	Notes
1	When resetting up a cache, all the schema documents specified in the schema definition file are parsed, and grammar objects are setup, so the parsing process takes more time. Therefore, the first parser after executing the <code>cacheon</code> command will take more than normal time for processing.
2	When resetting up a cache and starting or stopping J2EE servers, the directories with the names same as of the J2EE server names that exist below the directories specified in <code>diskcache_path</code> will be deleted. Therefore, do not generate files or directories with the names same as that of the J2EE servers in the directory specified in <code>diskcache_path</code> .
3	After setting up a cache, if you change the directory specifications of <code>diskcache_path</code> in the schema definition file, the disk cache might not be deleted when stopping the J2EE server or executing the <code>cacheoff</code> and <code>cacheon</code> commands. In such cases, manually delete the remaining files.

6.19 Notes on the high-speed parse support function

The following subsections provide notes on the high-speed parse support function:

6.19.1 Notes on setting up the preparsed object

Table 6–35: Notes on the high-speed parse support function (setting up the preparsed object)

No.	Notes
1	The enabling or disabling of the namespace when pre-parsing must match the enabling or disabling of the namespace when parsing. If these differ, the preparsing might be invalid.
2	When resolving an entity reference, the results of resolving by the entity resolver when pre-parsing must match the results of resolving by the entity resolver when parsing. If these results differ, the high speed effect might not be achieved.

6.19.2 Notes on the XML parser using the preparsed object

Table 6–36: Notes on the high-speed parse support function (XML parser using the preparsed object)

No.	Notes
1	The high speed effect might not be achieved when the following features of the parser that use the preparsed object are set up as false: 1. <code>http://xml.org/sax/features/external-general-entities</code> 2. <code>http://xml.org/sax/features/external-parameter-entities</code>

6.19.3 Notes on XML1.1

Table 6–37: Notes on the high-speed parse support function (XML1.1)

No.	Notes
1	The high-speed parse support function does not support XML1.1. Therefore, an error occurs if pre-parsing for an XML1.1 document. The high-speed parse support function is not applicable even if an XML1.1 document is set up as a target for performing the high-speed parse support function.

6.19.4 Notes on XInclude (high-speed parse support function)

Table 6–38: Notes on the high-speed parse support function (XInclude)

No.	Notes
1	The high-speed parse support function is not used to pre-parse a file fetched by XInclude.

6.19.5 Notes on how to specify an absolute path

Table 6–39: Notes on the high-speed parse support function (how to specify an absolute path)

No.	Notes
1	Use a forward slash (/) as a delimiter in the path to specify an absolute path.
2	<p>In Windows, include the drive name in the absolute path.</p> <p>The specification examples of the absolute path in Windows and UNIX are as follows:</p> <p>In Windows</p> <pre>D:/home/xml/training/training1.xml</pre> <p>In UNIX</p> <pre>/home/xml/training/training1.xml</pre>

6.19.6 Notes on the Pre-Parse XML document

Table 6–40: Notes on the high-speed parse support function (Pre-Parse XML document)

No.	Notes
1	There is an upper limit for the length of the element names and attribute names that can be coded in the Pre-Parse XML document. If the length [#] obtained by adding the number of characters of the longest element name and attribute name coded in the Pre-Parse XML document, exceeds 21,000 characters, the KECX09607-E error occurs, and the process might be interrupted. For details about the message, see <i>11. KECX (Messages Output by Cosminexus XML Processor)</i> in the manual <i>uCosminexus Application Server Messages</i> .

#

For example, in the following XML document, the number of characters of the longest element name (`element` or `content`) is 7, and the number of characters of the longest attribute name (`index`) is 5. Therefore, the number of characters obtained by adding these will be 12.

```
<root>
<data index="1"/>
<element>
  <content>example</content>
</element>
</root>
```

6.19.7 Notes on tuning information

Table 6–41: Notes on the high-speed parse support function (tuning information)

No.	Notes
1	Use a forward slash (/) as a delimiter in the path of the system identifier of the XML document to be parsed.
2	<p>If the File object created by specifying a relative path in the constructor argument, is specified in the <code>newPreparedObject</code> method of the <code>PreparedObjectFactory</code> class, the path delimiting characters of the file name of Pre-Parse XML document that are output to the tuning summary file and details file, are duplicated and output as follows:</p> <pre>Prepared XML: C:\\a.xml</pre>

6.19.8 Notes on parse performance

Table 6–42: Notes on the high-speed parse support function (Parse performance)

No.	Notes
1	If you apply the high-speed parse support function to an XML document containing an external entity reference, the parse performance might not improve as compared to the regular parsing. Evaluate the performance beforehand to apply the high-speed parse support function to an XML document containing an external entity reference.

6.20 Notes on JAXB

The following subsections give cautionary notes on JAXB:

6.20.1 Common notes

The following table describes the common notes.

Table 6–43: Notes on JAXB (common)

No.	Notes
1	If you apply the <code>getMessage</code> method to the exception object thrown by the method provided in the JAXB specifications, the return value will become <code>null</code> , and you might not be able to acquire the detailed message. In this case, if you obtain the wrapped exception object by applying the <code>getCause</code> method to the exception object and then apply the <code>getMessage</code> method, you may obtain the detailed message. You can obtain the detailed message by applying the <code>toString</code> method for <code>JAXBException</code> exception and its subclass.
2	<p>In the JAXB specifications, the operation of the methods, schema compiler, and schema generator might not be defined clearly at some locations. In such cases, the operations depend on the implementation. Also, if <code>null</code> is specified in the argument, and <code>Javadoc</code> describes that an exception other than the <code>NullPointerException</code> exception will occur, but in some cases the <code>NullPointerException</code> exception might occur. Make sure that there is no such situation where the operations that are not regulated are executed by prior checking of arguments.</p> <p>(Example 1) An exception of a type that is not specified in the exception column of <code>Javadoc</code> (The operation when <code>null</code> is specified in the argument is not regulated and hence <code>NullPointerException</code> occurs) occurs.</p> <p>(Example 2) The operations are not regulated when marshal or unmarshal is performed using the Java source generated when the operations of the schema compiler are not regulated.</p>
3	You cannot use the ISO-10646-UCS-4 encoding.
4	You cannot use an XML document of version 1.1.
5	You cannot use Unicode supplementary characters.
6	<p>Java SE 6 does not support JAXB 2.2 by default. Therefore, if you want to use JAXB 2.2 functions on a non-Hitachi JavaVM, specify <i>Cosminexus-installation-destination</i>/<code>jaxp/lib/csmjaxb-api.jar</code> using the Java endorsed standards override mechanism. For details on how to specify the settings, see the following Oracle website:</p> <p>http://download.oracle.com/javase/6/docs/technotes/guides/standards/</p>
7	A user application cannot include a JAXB1 implementation.

6.20.2 Notes on schema compiler

The following subsections give cautionary notes on schema compiler.

Do not process the schema corresponding to the notes described in item no. 1 to item no. 28 by the schema compiler. For item no. 29 to item no. 44, use the schema compiler after considering the notes.

Table 6–44: Notes on JAXB (schema compiler)

No.	Notes
1	<p>[Conditions]</p> <p>When all of the following conditions are applicable to contents model with iterative specifications:</p> <ul style="list-style-type: none">• An element definition in which <code>nillable="true"</code> is specified exists.• The contents model is not <code>mixed</code> and does not include a wildcard.

No.	Notes
	<ul style="list-style-type: none"> • The element definition of the contents model does not include an <code>xml</code> data type of <code>xs:list</code> and <code>xs:IDREF</code>, or a data type derived from <code>xs:list</code> and <code>xs:IDREF</code>. • Two different element definitions bound to the same Java data type do not exist for all element definitions of the contents model. <p>(Example)</p> <pre><xs:complexType> <xs:sequence maxOccurs="unbounded"> <xs:element name="A" type="xs:string" /> <xs:element name="B" type="xs:int" nillable="true" /> <xs:element name="C" type="xs:float" nillable="true"/> <xs:element name="D" type="xs:double" /> </xs:sequence> </xs:complexType></pre> <p>Operation of Cosminexus XML Processor</p> <p>A collection is optimized. The collection annotated with <code>@XmlElement</code>s instead of <code>@XmlElementRefs</code>/<code>@XmlElementRef</code> is output. An invalid operation might be performed when <code>unmarshal</code> and <code>marshal</code> is executed using the java source that is output.</p>
2	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. A reserved device name of Windows, such as <code>www.com1.hitachi</code> is included in the <code>targetNamespace</code> attribute of the <code>xs:schema</code> element. 2. The schema compiler is executed on Windows. <p>Operations of Cosminexus XML Processor</p> <p>Error occurs.</p> <p>(Example)</p> <pre>[ERROR] .\hitachi\com1\ObjectFactory.java (The specified path is not found. [errno=3, syscall=CreateFileW])</pre>
3	<p>[Conditions]</p> <p>When all of the following conditions are applicable to the first character of the character string that separates the URI specified in the <code>targetNamespace</code> attribute of the <code>xs:schema</code> element by a colon (:) or forward slash (/):</p> <ul style="list-style-type: none"> • The return value of <code>Character.isJavaIdentifierStart</code> becomes false. • The return value of <code>Character.isJavaIdentifierPart</code> becomes false. <p>Operations of Cosminexus XML Processor</p> <p>An error indicating an invalid package name occurs.</p> <p>(Example)</p> <pre>[ERROR] The package name 'publicid.__-.w3c.dtd_html_4_01.en' used for this schema is not a valid package name.</pre>
4	<p>[Conditions]</p> <p>When the URI specified in the <code>targetNamespace</code> attribute and <code>xmlns</code> attribute of the <code>xs:schema</code> element is configured only from the following characters:</p> <ul style="list-style-type: none"> • Periods (.) • Forward slashes (/) • Character string "www" <p>Operations of Cosminexus XML Processor</p> <p>Exceptions that are not provided such as <code>java.lang.ArrayIndexOutOfBoundsException</code> might occur.</p>
5	<p>[Conditions]</p> <p>When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> • When multiple <code>xs:any</code> elements exist in the contents model without iterative specification. • Using the complex type extension, both the base type and extension type include the <code>xs:any</code> type as the element contents. <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	<p>An error occurs by trying to generate multiple properties and not a single general contents property. For details about the general contents properties, see the <i>subsection 6.12.3 of JSR 222 The Java Architecture for XML Binding 2.1</i>.</p> <p>(Example)</p> <pre>[ERROR] Property "Any" is already defined. Use <jaxb:property> to resolve this conflict.</pre> <p>Note that if you specify <code>jaxb:property</code> custom binding and execute unmarshalling or marshalling using the java source with multiple <code>@XmlAnyElement</code> annotated properties that are output, some invalid operations might be performed.</p>
6	<p>[Conditions]</p> <p>When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> When the format is derived, <code>true</code> is specified for the value of the <code>mixed</code> attribute in a <code>complexType</code> element having the derived type. When the format is derived, <code>true</code> is specified for the value of the <code>mixed</code> attribute having the base type. <p>Operations of Cosminexus XML Processor</p> <p>The method for acquiring a child element of the derived format will not be generated.</p>
7	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> The <code>jaxb:typesafeEnumClass</code> custom binding that has <code>jaxb:typesafeEnumMember</code> as a child element for the <code>xs:simpleType</code> element. Also, the <code>name</code> attribute is specified in <code>jaxb:typesafeEnumMember</code>. The <code>jaxb:typesafeEnumMember</code> custom binding is specified for the <code>xs:enumeration</code> element that is a lower element of <code>xs:simpleType</code> specified in condition 1, and the <code>name</code> attribute is also specified. The attribute values of the <code>name</code> attribute specified in condition 1. and 2. differ. <p>Operations of Cosminexus XML Processor</p> <p>The enumeration constant name of the Java class is generated based on the <code>name</code> attribute specified in condition 1.</p>
8	<p>[Conditions]</p> <p>When custom binding is specified in any of the following XML schema elements:</p> <ul style="list-style-type: none"> <code>jaxb:typesafeEnumClass</code> is specified in the <code>xs:attributeGroup</code> element. <code>jaxb:typesafeEnumClass</code> is specified in the <code>xs:extension</code> element that is the child element of the <code>xs:complexContent</code> element. <code>jaxb:typesafeEnumClass/jaxb:typesafeEnumMember/jaxb:property</code> is specified in the <code>xs:restriction</code> element that is the child element of the <code>xs:simpleType</code> element or in the <code>xs:maxExclusive/xs:maxInclusive/xs:minExclusive/xs:maxLength/xs:pattern</code> facet of the child element. <code>jaxb:schemaBindings</code> is specified in the <code>xs:totalDigits/xs:whiteSpace</code> facet of the child element of <code>xs:restriction</code> element. <p>Operations of Cosminexus XML Processor</p> <p>The operations during the execution of JAXB when such a schema document is used are not regulated.</p>
9	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ul style="list-style-type: none"> The <code>jaxb:schemaBindings</code> custom binding in which the <code>map</code> attribute is <code>false</code> exists. The <code>jaxb:class</code> or <code>jaxb:typesafeEnumClass</code> custom binding exists. <p>Operations of Cosminexus XML Processor</p> <p>An error indicating invalid custom binding is output.</p> <p>(Example)</p> <pre>[ERROR] compiler was unable to honor this class customization. It is attached to a wrong place, or its inconsistent with other bindings.</pre>
10	<p>[Conditions]</p> <p>When the <code>jaxb:class</code> custom binding is specified for the local element definition (<code>xs:element</code> containing <code>=xs:complexType</code> as the child element) whose type is defined as an anonymous complex type.</p> <p>Operations of Cosminexus XML Processor</p> <p>An error is output.</p> <p>(Example)</p>

No.	Notes
	Exception in thread "main" java.lang.IllegalArgumentException: Illegal class inheritance loop. Outer class <i>element-name</i> may not subclass from inner class: <i>element-name</i>
11	<p>[Conditions] When the <code>jaxb:property</code> custom binding is specified for <code>xs:choice</code>, <code>xs:sequence</code>, or <code>xs:all</code> specified in the complex type definition <code>xs:restriction</code>.</p> <p>Operations of Cosminexus XML Processor An error is output.</p> <p>(Example) [ERROR] compiler was unable to honor this property customization. It is attached to a wrong place, or its inconsistent with other bindings.</p>
12	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The <code>jaxb:typesafeEnumClass</code> custom binding is specified in the <code>xs:complexType</code> element. 2. <code>true</code>, <code>1</code>, or <code>0</code> is specified in the <code>map</code> attribute specified in the custom binding of condition 1. <p>Operation of Cosminexus XML Processor The message <code>KECX06051-E cvc-enumeration-valid: Value 'true' is not facet-valid with respect to enumeration '[false]'</code>. It must be a value from the enumeration. <code>occurs</code>. The value <code>true</code> specified in the <code>map</code> attribute is correct, but the specification of the <code>jaxb:typesafeEnumClass</code> custom binding in the <code>xs:complexType</code> element itself is invalid.</p>
13	<p>[Conditions] When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> • When any the following is specified in the <code>collectionType</code> attribute of <code>jaxb:property</code>: <ul style="list-style-type: none"> - An empty character string - A class in which <code>java.util.List</code> is not implemented - A class in which <code>java.util.List</code> is implemented, but not a completely modified class • When any the following is specified in the <code>collectionType</code> attribute of <code>jaxb:globalBindings</code> <ul style="list-style-type: none"> - A class in which <code>java.util.List</code> is not implemented - A class in which <code>java.util.List</code> is implemented, but the class is not completely modified <p>Operations of Cosminexus XML Processor The java class is generated without an error occurrence.</p>
14	<p>[Conditions] When specifying an empty character string is specified in the <code>collectionType</code> attribute of <code>jaxb:globalBindings</code>.</p> <p>Operations of Cosminexus XML Processor Throws the exception <code>java.lang.StringIndexOutOfBoundsException</code>.</p>
15	<p>[Conditions] When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Specify the following two attributes in the <code>jaxb:class</code> custom binding: <ul style="list-style-type: none"> - <code>ref</code> attribute - <code>exclusive name</code> attribute • Specify the following two attributes in the <code>jaxb:class</code> custom binding: <ul style="list-style-type: none"> - <code>ref</code> attribute - <code>exclusive implClass</code> attribute • Specify the following two attributes in the <code>jaxb:typesafeEnumClass</code> custom binding: <ul style="list-style-type: none"> - <code>ref</code> attribute - <code>exclusive name</code> attribute • Specify the following two attributes in the <code>jaxb:typesafeEnumClass</code> custom binding: <ul style="list-style-type: none"> - <code>ref</code> attribute - <code>exclusive map</code> attribute

No.	Notes
	<p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. In such cases, the operations for executing JAXB are not defined.</p>
16	<p>[Conditions]</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> Specify the <code>jaxb:class</code> custom binding in the <code>xs:element</code> global element definition or the <code>xs:element</code> local element definition. Specify the <code>implClass</code> attribute in the <code>jaxb:class</code> custom binding. <p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. In such cases, the operations for executing JAXB are not defined.</p>
17	<p>[Conditions]</p> <p>When all the following condition are satisfied:</p> <ul style="list-style-type: none"> Specify the <code>jaxb:typesafeEnumClass</code> custom binding in the <code>xs:simpleType</code> definition of an enumerated facet. Specify <code>True</code>, <code>1</code>, or <code>0</code> in the <code>map</code> attribute of <code>jaxb:typesafeEnumClass</code> <p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. When the <code>map</code> attribute is not specified, the operations with the specified <code>true</code> or <code>1</code> will be executed by default.</p>
18	<p>[Conditions]</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> Specify the <code>jaxb:property</code> custom binding for a local or a global attribute declaration Specify <code>True</code>, <code>false</code>, <code>1</code>, or <code>0</code> in the <code>generateElementProperty</code> attribute. <p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. Java source including the property is output.</p>
19	<p>[Conditions]</p> <p>When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> Specify the <code>jaxb:property</code> custom binding for multiple element declarations with different <code>name</code> attributes in the same format. Specify <code>False</code> in the <code>generateElementProperty</code> attribute <p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. Java source including <code>JAXBElement</code> is output.</p>
20	<p>[Conditions]</p> <p>When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> Specify a class that does not exist in the same inheritance hierarchy as the default base type is specified, in the <code>name</code> attribute of the <code>jaxb:baseType</code> element. Specify a class in which the Java class name is not fully modified, in the <code>name</code> attribute of the <code>jaxb:baseType</code> element. <p>Operations of Cosminexus XML Processor</p> <p>An error does not occur. Java source is output.</p>
21	<p>[Conditions]</p> <p>When the <code>attachmentRef</code> attribute of the <code>jaxb:property</code> custom binding is to be used.</p> <p>Operations of Cosminexus XML Processor</p> <p>The following error is output during the schema compilation:</p> <pre>[ERROR] KECX06031-E cvc-complex-type.3.2.2: Attribute 'attachmentRef' is not allowed to appear in element 'jaxb:property'.</pre>
22	<p>[Conditions]</p> <p>When specifying character strings other than <code>asWordSeparator</code> or <code>asCharInWord</code>, in the <code>underscoreBinding</code> attribute of the <code>jaxb:globalBindings</code> custom binding.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>NullPointerException</code> exception is thrown.</p>

No.	Notes
23	<p>[Conditions] When specifying skipGeneration in the typesafeEnumMemberName attribute of the jaxb:globalBindings custom binding.</p> <p>Operations of Cosminexus XML Processor The following error is output: KECX06051-E cvc-enumeration-valid: Value 'skipGeneration' is not facet-valid with respect to enumeration '[generateError, generateName]'. It must be a value from the enumeration.</p> <p>Workaround When the typesafeEnumMemberName attribute is not specified, the default operation changes to the skipGeneration operation.</p>
24	<p>[Conditions] When specifying and customizing the jaxb:globalBindings custom bindings multiple times:</p> <ul style="list-style-type: none"> • The xmlType attributes have different name attribute values, and the values of the xmlType attributes have same jaxb:javaType • jaxb:serializable with different uid attribute values <p>Operations of Cosminexus XML Processor In such cases, the operations for executing JAXB are not defined.</p>
25	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Specify primitive in the optionalProperty attribute of the jaxb:globalBindings custom binding. • Anyone of the following local element exists in the schema: <ul style="list-style-type: none"> - Primitive type element with the specified minOccurs="0". - Primitive type element defined in xs:choice <p>Operations of Cosminexus XML Processor Output in wrapper class type and not the primitive type.</p>
26	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Specify False in the generateElementProperty attribute of jaxb:globalBindings custom binding. • The specifications that become the property of the JAXBElement type, such as multiple elements with the same type and different names exist in the schema. <p>Operations of Cosminexus XML Processor An error does not exist. In such case, the operations for executing JAXB are not defined.</p>
27	<p>[Conditions] When specifying jaxb:inlineBinaryData in xs:complexType having simple contents of the binary type.</p> <p>Operations of Cosminexus XML Processor An invalid custom binding error is output. Example: [ERROR] compiler was unable to honor this inlineBinaryData customization. It is attached to a wrong place, or its inconsistent with other bindings.</p>
28	<p>[Conditions] Anyone of the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Specify a non existing java data type in the name attribute of jaxb:javaType. • Specify the parseMethod attribute or the printMethod attribute of jaxb:javaType, and the primitive type in the name attribute. • In the name attribute of jaxb:javaType, specify the java data type for which a constructor with one argument of the String type is not defined. <p>Operations of Cosminexus XML Processor An error does not occur.</p>

No.	Notes
29	The standard specifications provide the methods for customizing the contents of the documentation comments of Java classes generated using the <code>jaxb:javadoc</code> custom binding. Even when <code>jaxb:javadoc</code> is not specified, the default documentation comments are generated, but the contents are not provided in the standard specifications.
30	<p>[Conditions]</p> <p>When there is a simple type definition where the range specified in any of the <code>xs:totalDigits</code>, <code>xs:maxExclusive</code>, <code>xs:maxInclusive</code>, <code>xs:minExclusive</code>, or <code>xs:minInclusive</code> facets is within the <code>int</code> or <code>long</code> range.</p> <p>Operations of Cosminexus XML Processor</p> <p><code>java.math.BigInteger</code> is generated always when the <code>xs:totalDigits</code> facet is used.</p> <p>When the <code>xs:maxExclusive</code>, <code>xs:maxInclusive</code>, <code>xs:minExclusive</code>, or <code>xs:minInclusive</code> facet is used, the Java type is generated according to the algorithm specified in the <i>subsection 6.2.2 of JSR 222 The Java Architecture for XML Binding 2.1</i>.</p>
31	<p>[Conditions]</p> <p>When a model group definition containing any of the following exists:</p> <ul style="list-style-type: none"> • An anonymous complex type definition • <code>xs:enumeration</code> facet definition mapped in <code>enum</code> type • The contents model mapped in the element class <p>Operations of Cosminexus XML Processor</p> <p>The model group name is not added to the created class name.</p>
32	<p>[Conditions]</p> <p>When the <code>jaxb:class</code> custom binding is specified in the <code>xs:element</code> local element definition.</p> <p>Operations of Cosminexus XML Processor</p> <p>Inherits <code>JAXBElement<T></code> and creates element factory method that returns the created classes.</p> <p>(Example)</p> <pre>public class ObjectFactory { public Base.Bar createBaseBar(String value) { return new Base.Bar(value); } }</pre> <p>Note</p> <p><code>Base.Bar</code> is the class inheriting <code>JAXBElement<T></code>.</p>
33	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The <code>xs:element</code> global element definition exists. 2. The value of the <code>abstract</code> attribute specified in condition 1. is <code>true</code>. <p>Operations of Cosminexus XML Processor</p> <p>The element factory method corresponding to the element definition is generated, but instances cannot be generated using the factory method.</p>
34	<p>[Conditions]</p> <p>When either of the following is applicable to the schema document:</p> <ul style="list-style-type: none"> • Element definitions with the same element name and different name space are specified in the contents model. • Two element definitions with element names <code>class</code> or <code>Class</code> and <code>clazz</code> or <code>Clazz</code> are specified in the contents model. <p>Operations of Cosminexus XML Processor</p> <p>The name conflicting error does not occur, and the process is performed as a general contents list annotated by <code>@XmlElementRef</code>/<code>@XmlElementRefs</code>. For details about the contents list, see the <i>subsection 6.12.3 of JSR 222 The Java Architecture for XML Binding 2.1</i>.</p>
35	<p>[Conditions]</p> <p>When either <code>lax</code> or <code>skip</code> is specified in the <code>processContents</code> attribute of <code>xs:any</code>.</p> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	The data type of the bind property will be the <code>java.lang.Object</code> type when the <code>processContents</code> attribute is <code>lax</code> , and the <code>org.w3c.dom.Element</code> type when the <code>processContents</code> attribute is <code>skip</code> .
36	<p>[Conditions] When the <code>fixed</code> attribute of the <code>xs:element</code> element is specified.</p> <p>Operations of Cosminexus XML Processor The value of the specified <code>fixed</code> attribute is not applied to the <code>defaultValue</code> element of <code>@XmlElement</code> and <code>@XmlElementDecl</code> of the Java source that is generated.</p> <p>[Corrective action] To apply the value of the <code>fixed</code> attribute as a default value, specify the <code>default</code> attribute instead of the <code>fixed</code> attribute.</p>
37	<p>[Conditions] When a character string is specified in the <code>value</code> attribute of the <code>xs:enumeration</code> element in such a way so that there are conflicts in the generated java constant identifier.</p> <p>(Example) <code>name-with-dashes</code> and <code>name_with_dashes</code></p> <p>Operations of Cosminexus XML Processor The enumeration definition is not output. No error occurs.</p>
38	<p>[Conditions] When all of the following conditions are applicable to the URI specified in the <code>targetNamespace</code> attribute of the <code>xs:schema</code> element:</p> <ul style="list-style-type: none"> • Top-level domains (such as <code>com</code>, <code>gov</code>, <code>net</code>, <code>org</code>, and <code>edu</code>) are not included. • The country code is not included. <p>Operations of Cosminexus XML Processor The domain name of the package name of generated java source will be in the reverse order, and <code>www.</code> will be deleted.</p>
39	<p>[Conditions] When any of the following is specified in the element name and attribute name:</p> <ul style="list-style-type: none"> • java keyword • java literal (<code>true</code>, <code>false</code>, or <code>null</code>) • Method names defined in the <code>java.lang.Object</code> class • Method names defined in the <code>javax.xml.bind</code> class <p>Operations of Cosminexus XML Processor The field and method name will be generated without the occurrence of a name conflicting error. A field name with an underscore" <code>_</code>" will be created for java keyword and java literal.</p>
40	<p>[Conditions] When a redefined complex type definition exists.</p> <p>Operations of Cosminexus XML Processor For a redefined complex type definition, a class name with the prefix <code>Original</code> is generated instead of a class name with attached <code>_</code>.</p>
41	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • A model group definition exists. • Specify the <code>suffix</code> attribute or the <code>prefix</code> attribute of <code>jaxb:modelGroupName</code> in the <code>jaxb:schemaBindings</code> custom binding. <p>Operations of Cosminexus XML Processor The model group definitions are not mapped in a class, so the specification of <code>jaxb:modelGroupName</code> is not applied to the generated source.</p>
42	<p>[Conditions] When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. Specify <code>asCharInWord</code> in the <code>underscoreBinding</code> attribute of <code>jaxb:globalBindings</code> custom binding.

No.	Notes
	<p>2. Underline is included in the name attribute of <code>xs:element</code> or in the name attribute <code>xs:attribute</code> with the specified fixed attribute.</p> <p>3. For the name attribute of <code>xs:attribute</code> in step 2, True is specified in the <code>fixedAttributeAsConstantProperty</code> attribute of <code>jaxb:globalBindings</code>.</p> <p>Operations of Cosminexus XML Processor A Java identifier different from the specifications is generated. Example:</p> <ul style="list-style-type: none"> For the name attribute of <code>xs:element</code>: <code>child_element => getChild_Element</code> (correct is <code>getChild_element</code>) For the name attribute of <code>xs:attribute</code>: <code>child_attribute => CHILD___ATTRIBUTE</code> (correct is <code>CHILD_ATTRIBUTE</code>)
43	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> In the <code>enableJavaNamingConventions</code> attribute of the <code>jaxb:globalBindings</code> custom binding, specify true or do not specify the attribute. In the name attribute of <code>jaxb:package</code> that is the child element of <code>jaxb:schemaBindings</code>, specify an invalid value as a java identifier. <p>Operations of Cosminexus XML Processor The conversion into an appropriate java identifier is not executed.</p>
44	<p>[Conditions] When specifying <code>jaxb:serializable</code> without the <code>uid</code> attribute, in the <code>jaxb:globalBindings</code> custom binding.</p> <p>Operations of Cosminexus XML Processor An error does not occur. In such cases, the operations for executing JAXB are not defined.</p>

6.20.3 Notes on schema generator

The following subsections give cautionary notes on schema generator.

Do not process the Java sources corresponding to the notes described in item no. 1 to item no. 32, 56, 57, 59 in the schema generator. For item no. 33 to item no. 55, and item no. 58, 60, 61 use the schema generator after considering the notes.

Table 6–45: Notes on JAXB (schema generator)

No.	Notes
1	<p>The scope for specification of JAXB mapping annotation has been restricted in the JAXB specifications. As described in the following example, an error might not occur when the JAXB mapping annotation is specified at a location that is outside the scope of specification described in the JAXB specifications, so take precautions. In such cases, the operations for executing JAXB are not defined.</p> <p>Example:</p> <ul style="list-style-type: none"> The JAXB mapping annotation that cannot be specified in a package is specified in the package. <code>@XmlRootElement</code> and <code>@XmlType</code> are specified in a class that is not a top level class. <code>@XmlEnumValue</code> is specified in a field. <code>@XmlRegistry</code> is specified in the enumeration type. JAXB mapping annotation, which cannot be specified simultaneously as per the specifications, are specified simultaneously.
2	<p>[Conditions] When any of the following combinations of JAXB mapping annotations are specified simultaneously:</p> <ul style="list-style-type: none"> <code>@XmlAttribute</code> and <code>@XmlValue</code> <code>@XmlAttribute</code> and <code>@XmlMimeType</code>

No.	Notes
	<ul style="list-style-type: none"> • <code>@XmlElement</code> and <code>@XmlValue</code> <p>Operations of Cosminexus XML Processor An error is output. No schema is generated.</p> <p>(Example) For <code>@XmlAttribute</code> and <code>@XmlValue</code> <code>mypackage.ChildType#name</code> has mutually exclusive annotations <code>@javax.xml.bind.annotation.XmlAttribute</code> and <code>@javax.xml.bind.annotation.XmlValue</code></p> <p>For <code>@XmlAttribute</code> and <code>@XmlMimeType</code> <code>javax.xml.bind.annotation.XmlMimeType</code> annotation cannot be placed here</p>
3	<p>[Conditions] When any one of the following is specified in a transient field</p> <ul style="list-style-type: none"> • <code>@XmlAttribute</code> • <code>@XmlElement</code> • <code>@XmlElements</code> • <code>@XmlElementWrapper</code> • <code>@XmlList</code> • <code>@XmlMimeType</code> • <code>@XmlValue</code> • <code>@XmlAnyElement</code> <p>Operations of Cosminexus XML Processor Error occurs.</p> <p>(Example) Transient field "child" cannot have any JAXB annotations.</p>
4	<p>[Conditions] When any one of the following is specified in a static field:</p> <ul style="list-style-type: none"> • <code>@XmlElement</code> • <code>@XmlElements</code> • <code>@XmlElementWrapper</code> • <code>@XmlList</code> • <code>@XmlMimeType</code> • <code>@XmlValue</code> • <code>@XmlAnyElement</code> • <code>@XmlID</code> • <code>@XmlIDREF</code> <p>Operations of Cosminexus XML Processor No error occurs.</p>
5	<p>[Conditions] When not even a single <code>@XmlElement</code> is specified in <code>@XmlElements</code>.</p> <p>Operations of Cosminexus XML Processor An invalid schema is output.</p>
6	<p>[Conditions] When the <code>@XmlJavaTypeAdapters</code> annotation that does not contain even a single <code>@XmlJavaTypeAdapter</code> is specified.</p> <p>Operations of Cosminexus XML Processor A schema is output as if <code>@XmlJavaTypeAdapters</code> is not specified.</p>
7	<p>[Conditions] When <code>@XmlSeeAlso</code> with an empty value element array is specified.</p> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	A schema such as when <code>@XmlSeeAlso</code> is not specified is output.
8	<p>[Conditions] When an empty character string is specified in the name element of any of <code>@XmlElementWrapper</code>, <code>@XmlAttribute</code>, <code>@XmlElement</code>, <code>@XmlRootElement</code>, or <code>@XmlElementDecl</code>.</p> <p>Operations of Cosminexus XML Processor</p> <ul style="list-style-type: none"> For <code>@XmlElementWrapper</code> or <code>@XmlElement</code>, the element name is generated from the field name and property name, in the same way as when the name element is not specified. For <code>@XmlAttribute</code>, an invalid attribute definition called <code><xs:attribute name="" .../></code> is generated. For <code>@XmlRootElement</code> or <code>@XmlElementDecl</code>, an invalid element definition called <code><xs:element name="" .../></code> is generated.
9	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> <code>@XmlElement</code> is added to a field or JavaBean property of <code>collection</code> type with parameters. Alternatively, <code>@XmlElements</code> is added to the JavaBean property of <code>array</code> type or <code>list</code> type with parameters. The type element of <code>@XmlElement</code> indicated in condition 1. is specified explicitly. <p>Operations of Cosminexus XML Processor No constraints are applicable to the type element of <code>@XmlElement</code>. The operations for the JAXB execution when such a Java source is used are not regulated. For details about the constraints, see the <i>subsection 8.9.1.2 and Javadoc of JSR 222 The Java Architecture for XML Binding 2.1</i>.</p>
10	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> <code>@XmlElements</code> is specified in the field or JavaBean property of <code>collection</code> type with parameters. Multiple <code>@XmlElements</code> are specified in the <code>@XmlElements</code> elements of condition 1. <p>Operations of Cosminexus XML Processor No error occurs. However, the processing will not be performed properly even when <code>unmarshal</code> is executed using the schema and java source.</p> <p>[Corrective action] If you want to specify multiple element definitions in a property or JavaBean field of <code>collection</code> type with parameters, use <code>@XmlElementRefs</code> and <code>@XmlElementRef</code>, and process them as <code>collection</code> type of <code>JAXBElement</code>.</p>
11	<p>[Conditions] When an invalid value is specified for the type of a field or JavaBean property in which <code>@XmlElement</code> is specified in the <code>defaultValue</code> element of <code>@XmlElement</code>.</p> <p>Operations of Cosminexus XML Processor A schema document is generated for an element definition in which an invalid value is specified for the <code>default</code> attribute, and no error occurs. However, when you perform <code>unmarshal</code> with an <code>Unmarshaller</code> in which the generated schema is set up, an error indicating invalid schema (KECX06161-E) occurs. The generated schema will become invalid.</p>
12	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> A value other than a null character string is specified in the namespace element of <code>@XmlSchema</code>. <code>@XmlNs</code> is specified in the <code>xmlns</code> element of <code>@XmlSchema</code>. The value of a <code>prefix</code> element of <code>@XmlNs</code> specified in condition 2. is different from <code>tns</code>, and the value of a namespace element is different from the value of condition 1. <p>Operations of Cosminexus XML Processor No error message is output and no schema can be generated.</p>
13	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> <code>@XmlAttribute</code> is added to a field or JavaBean property of <code>primitive</code> type. Either the <code>required</code> element is not specified in <code>@XmlAttribute</code> indicated in condition 1., or <code>false</code> is specified. <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	The value of use attribute of a generated attribute definition is always required.
14	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. A class mapped to the simple type list exists. 2. @XmlList is specified in a List instance that contains the class specified in condition 1. as the type parameter. <p>Operations of Cosminexus XML Processor No error occurs, and a schema in which @XmlList is applied will be generated.</p>
15	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The @XmlRootElement annotation, and @XmlType annotation in which the name element is an empty character string, are added to the class. 2. The type of the field or JavaBean property of the class specified in condition 1. is the class itself that is specified in condition 1. <p>Operations of Cosminexus XML Processor No error message is output and no schema can be generated.</p> <p>[Corrective action] Set up the name of the field or JavaBean specified in condition 2. to the same name as the element class name (however, name must be the converted name based on the rules for converting an XML name to Java name).</p> <p>(Example) <pre>@XmlRootElement @XmlType(name="") public class ItemType { public ItemType itemType;}</pre></p>
16	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. An abstract modifier is specified in a class mapped in complex type. 2. @XmlValue is added to the field or JavaBean property of the class specified in condition 1. 3. @XmlAttribute is added to the field or JavaBean property of the class specified in condition 1. <p>Operations of Cosminexus XML Processor The abstract attribute cannot be added to the generated xs:complexType element.</p>
17	<p>[Conditions] When @XmlEnum is specified in the enumeration type and the class that is not mapped in the simple type (example: java.lang.Object) is specified in the value element.</p> <p>Operations of Cosminexus XML Processor No error is output. The operations during the execution of JAXB when such a Java source is used are not regulated.</p>
18	<p>[Conditions] When @XmlEnumValue is specified in the enumeration constant, and inappropriate words and phrases are specified for the @XmlEnum.value() type, in the value element.</p> <p>Operations of Cosminexus XML Processor No error is output. The operations during the execution of JAXB when such a Java source is used are not regulated.</p>
19	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The @XmlRootElement annotation, and the @XmlType annotation in which the name element is an empty character string, are added to the class. 2. A class exists in such a way so that the type of the field or JavaBean property is the class itself that is specified in condition 1. 3. The element name mapped to the field or JavaBean property specified in condition 2. is different from the element name mapped to condition 1. <p>(Example) <pre>public class Root { // If the field name of condition 2. and 3. is childType, an element reference will be generated.</pre></p>

No.	Notes
	<pre>protected ChildType child; } // Class specified in condition 1. @XmlRootElement @XmlType(name = "") public class ChildType { public String fname; } </pre> <p>Operations of Cosminexus XML Processor An element definition (name attribute) will be generated.</p>
20	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The <code>@XmlRootElement</code> annotation, and the <code>@XmlType</code> annotation in which the name element is an empty character string, are added to the class. 2. A class with the type of the field or JavaBean property, same as in the class specified in condition 1 exists. 3. An element name mapped to the field or JavaBean property specified in condition 2. is same as the element name mapped to condition 1. 4. A value other than <code>\u0000</code> is specified in the <code>defaultValue</code> element of <code>@XmlElement</code> specified in the field or JavaBean property of condition 2. <p>(Example)</p> <pre>public class Root { @XmlElement(defaultValue="default") // Condition 4. protected ChildType childType; // Condition 2., 3. } // Class specified in condition 1. @XmlRootElement @XmlType(name = "") public class ChildType { public String fname; } </pre> <p>Operations of Cosminexus XML Processor An element definition (name attribute) containing the default attribute is generated.</p>
21	<p>[Conditions] When a JavaBean property is invalid because either of the <code>setter</code> method or <code>getter</code> method does not exist.</p> <p>Operations of Cosminexus XML Processor The schema is generated in such a way so that there is a correct JavaBean with both the <code>setter</code> and the <code>getter</code> method. An error does not occur. The operations during the execution of JAXB when such a Java source is used are not regulated.</p>
22	<p>[Conditions] When specifications exists in an element of the JAXB mapping annotation, in such a way so that an invalid schema document is generated.</p> <p>(Example)</p> <pre>@XmlRootElementpublic class ItemType { @XmlElement(name="A") public String item1; @XmlElement(name="A") public String item2; } </pre> <p>The same element name ("A") is specified in different fields of <code>@XmlElement (name="A")</code>.</p> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	No error check is performed, and an invalid schema document is generated.
23	<p>[Conditions] When the character string "http://www.w3.org/2001/XMLSchema" that indicates a name space URI specific to the XML Schema specifications is specified in the namespace element of the JAXB mapping annotation.</p> <p>Operations of Cosminexus XML Processor No schema document containing such name space URI in the targetNamespace attribute is generated. There is no purpose or meaning in specifying the name space URI.</p>
24	<p>[Conditions] When @XmlElement and @XmlMixed are specified simultaneously.</p> <p>Operation of Cosminexus XML Processor No error message is output and no schema can be generated.</p>
25	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The data type of a file and property that specify @XmlElementRef is JAXBElement. • @XmlElementRef.name() and @XmlElementRef.namespace() are same as the following JAXB mapping annotations that are the element factory classes as per the @XmlElementDecl annotation within the class that is annotated with @XmlRegistry: <ul style="list-style-type: none"> - @XmlElementDecl.name() - @XmlElementDecl.namespace() • Specify ##default in @XmlElementRef.namespace(). <p>Operations of Cosminexus XML Processor No error message is output and no schema can be generated.</p>
26	<p>[Conditions] When specifying the @XmlSchemaType annotation without the type element into a package.</p> <p>Operations of Cosminexus XML Processor An error does not occur. In the case of using such Java source, the operations for executing JAXB are not defined.</p>
27	<p>[Conditions] When annotated with multiple @XmlID in a class.</p> <p>Operations of Cosminexus XML Processor An error does not occur, and a schema is generated.</p>
28	<p>[Conditions] When any of the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The property with the attached @XmlID annotation is not included in the property of the collection type java.lang.Object type. • The property with the attached @XmlID annotation is not included in the java.lang.Object type property. • The field with the attached @XmlID annotation is not included in the property of the collection type java.lang.Object type. • The field with the attached @XmlID annotation is not included in the java.lang.Object type property. <p>Operations of Cosminexus XML Processor An error does not occur, and the schema is generated.</p>
29	<p>[Conditions] When coding the following that is in contradiction to the "There is only one JavaBean property annotated by @XmlAnyElement in a class and its super class" restriction:</p> <ul style="list-style-type: none"> • There are multiple fields and properties where @XmlAnyElement is specified in the class. • The field and property where @XmlAnyElement is specified respectively in certain class and its sub-class exists. <p>Operations of Cosminexus XML Processor No error is output. The operations during the execution of JAXB when such a Java source is used are not regulated.</p>
30	[Conditions]

No.	Notes
	<p>When <code>@XmlAttachmentRef</code> is added to the field or JavaBean property.</p> <p>Operations of Cosminexus XML Processor</p> <p>The value of the <code>minOccurs</code> attribute of the generated element definition will become 0.</p>
31	<p>[Conditions]</p> <p>When the following java types are mapped in the schema document: <code>char, Boolean, Character, Byte, Short, Integer, Long, Float, Double, byte[]</code></p> <p>Operations of Cosminexus XML Processor</p> <p>Mapping will be performed in the <code>xs:unsignedShort, xs:boolean, xs:unsignedShort, xs:byte, xs:short, xs:int, xs:long, xs:float, xs:double, xs:base64Binary</code> types respectively.</p>
32	<p>[Conditions]</p> <p>When the <code>@XmlJavaTypeAdapter</code> annotation that does not contain the <code>type</code> element in the package statement, is specified.</p> <p>Operations of Cosminexus XML Processor</p> <p>No error occurs, and a schema in which the specification of <code>@XmlJavaTypeAdapter</code> is enabled will be generated.</p>
33	<p>[Conditions]</p> <p>When <code>@XmlAttribute</code> is added to the static final field, or public static field.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>fixed</code> attribute cannot be added to the generated <code>xs:attribute</code> element.</p>
34	<p>[Conditions]</p> <p>When the <code>@XmlElement</code> annotation, in which the value of the <code>required</code> element is <code>false</code>, is specified in the field or JavaBean property of primitive type multi-dimensional array.</p> <p>Operations of Cosminexus XML Processor</p> <p>The value of the <code>minOccurs</code> attribute will become 0.</p>
35	<p>[Conditions]</p> <p>When any one of <code>@XmlList</code>, <code>@XmlJavaTypeAdapter</code>, <code>@XmlAttachmentRef</code>, or <code>@XmlMimeType</code> is added to the parameters of the method.</p> <p>Operations of Cosminexus XML Processor</p> <p>The specification of these JAXB mapping annotations will be ignored.</p>
36	<p>[Conditions]</p> <p>When the <code>@XmlSchema</code> annotation that does not contain the <code>location</code> element in the package, is specified.</p> <p>Operations of Cosminexus XML Processor</p> <p>The default value of the <code>location</code> element is interpreted as <code>##generate</code>.</p>
37	<p>[Conditions]</p> <p>When the <code>@XmlType</code> annotation is added to the enumeration type.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>final</code> attribute cannot be added to the generated <code>xs:simpleType</code>.</p>
38	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The name space URI is specified in the <code>namespace</code> element of <code>@XmlSchema</code>. 2. <code>@XmlNs</code> is specified in the <code>xmlns</code> element of <code>@XmlSchema</code>, and the name space URI is specified in the <code>namespaceURI</code> element. 3. The name space URI specified in condition 1. and 2. is the same character string. <p>Operations of Cosminexus XML Processor</p> <p>The name space declarations specified in condition 1. and condition 2. are generated separately. There will be no problem even if the generated schema is used.</p>
39	<p>[Conditions]</p> <p>When a class name or enumeration type name beginning with continuous upper-case characters exists.</p> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	The name obtained by converting the first continuous upper-case characters of the class name or enumeration type name to the lower-case characters is mapped to the XML name.
40	<p>[Conditions] When <code>true</code> is specified in the <code>nillable</code> element of <code>@XmlElementWrapper</code>.</p> <p>Operations of Cosminexus XML Processor <code>minOccurs="0"</code> is specified in the element definition of the generated schema.</p>
41	<p>[Conditions] When any of the following conditions is applicable:</p> <ul style="list-style-type: none"> • <code>@XmlElement</code> is added to the field or JavaBean property. • <code>@XmlElementWrapper</code> is added to the field or JavaBean property. • <code>@XmlRootElement</code> is added to the class or enumeration type. <p>Operations of Cosminexus XML Processor The <code>final</code> attribute and <code>block</code> attribute cannot be added to the <code>xs:element</code> element. Also, the <code>finalDefault</code> attribute and <code>block</code> attribute cannot be added to the <code>xs:schema</code> element.</p>
42	<p>[Conditions] When the <code>@XmlValue</code> annotation is added to the field or JavaBean property.</p> <p>Operations of Cosminexus XML Processor The <code>final</code> attribute cannot be added to the generated <code>xs:simpleType</code> element or <code>xs:complexType</code> element.</p>
43	<p>[Conditions] When executing the <code>csmschemagen</code> command in a directory whose absolute path includes a hash mark (#).</p> <p>(Example) <code>cd C:\jaxb\abc#xyz\test</code> <code>csmschemagen mypackage\TEST.java</code></p> <p>Operations of Cosminexus XML Processor The value of the <code>schemaLocation</code> attribute of the <code>xs:import</code> element of the generated schema document will become invalid.</p> <p>[Corrective action] Execute the <code>csmschemagen</code> command in a directory whose absolute path does not include a hash mark (#).</p>
44	<p>[Conditions] When an output destination directory is not specified in an argument of the <code>-d</code> option in the <code>csmschemagen</code> command.</p> <p>Example: <code>csmschemagen -d mypackage\TEST.java</code></p> <p>Operations of Cosminexus XML Processor The schema is not output. Moreover, an error might not be output.</p>
45	<p>[Conditions] In the property and fields, if the values to be mapped with different formats are specified in <code>type</code> of <code>@XmlElement</code> and <code>name</code> of <code>@XmlSchemaType</code>.</p> <p>Operations of Cosminexus XML Processor A schema document of the element definition with the data type specified in <code>name</code> of <code>@XmlSchemaType</code> is generated.</p>
46	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • In the field or the JavaBean property, <code>@XmlElement</code> of the <code>namespace</code> element that is different from the name space of the class is specified. • <code>@XmlID</code> or <code>@XmlIDREF</code> is specified in the field or the JavaBean property. <p>Operations of Cosminexus XML Processor The specified <code>@XmlID</code> or <code>@XmlIDREF</code> is not be applied in the generated global element definitions.</p>
47	<p>[Conditions] When other than <code>" "</code> is specified in the <code>substitutionHeadName</code> element of <code>@XmlElementDecl</code>.</p>

No.	Notes
	Operations of Cosminexus XML Processor The substitutionGroup attribute of the xs:element element is not output.
48	[Conditions] When a value other than \u0000 is specified in the defaultValue element of @XmlElementDecl. Operations of Cosminexus XML Processor The default attribute of the xs:element element is not be output.
49	[Conditions] When all the following conditions are satisfied: <ul style="list-style-type: none"> Specify @XmlAccessorType(XmlAccessorType.ALPHABETICAL) in a package and a class. In one class, there are multiple fields and properties in which @XmlAttribute annotation are not specified in the alphabetical order. Operations of Cosminexus XML Processor A schema, with the attributes sorted in the alphabetical order, is generated.
50	[Conditions] When specifying @XmlAccessorType and @XmlAccessorType in an internal class which is not of the top level class. Operations of Cosminexus XML Processor An error does not occur, and the annotation of the specification becomes valid.
51	[Conditions] When you specify a Java source with an encoding different from the system encoding Operations of Cosminexus XML Processor No error message is output and an invalid schema might be generated.
52	[Conditions] When all the following conditions are satisfied: <ol style="list-style-type: none"> @XmlElementRef is added to multiple field values or JavaBean properties. The required element is not specified for @XmlElementRef from 1., or true is specified in the required element. Operations of Cosminexus XML Processor The value of the minOccurs attribute is 0.
53	[Conditions] When all the following conditions are satisfied: <ul style="list-style-type: none"> An invalid encoding is specified in the -encoding option. Options other than the -encoding option are not specified. Operations of Cosminexus XML Processor Error is not output and the schema is not generated.
54	[Conditions] When any of the following conditions are satisfied: <ul style="list-style-type: none"> Write permission is not set for the current directory or the directory specified for the -d option. The value specified for the -d option is not a directory. Write permission is not set for the schema document that already exists in the current directory or the directory specified for the -d option. The -d or -encoding option is specified but no Java source file is specified. The version of Cosminexus is 09-70 or later. Operations of Cosminexus XML Processor The command ends without outputting an error message.

6.20.4 Notes on runtime

The following table describes the notes on runtime.

Table 6–46: Notes related to JAXB (Runtime)

No.	Notes
1	<p>When performing multi-thread programming, the objects provided in the packages with a name beginning with <code>javax.xml.bind</code> are not thread safe (as an exception, the <code>javax.xml.bind.JAXBContext</code> class is thread safe). Therefore, multiple threads must not access these objects at the same time. To avoid conflicts between threads, use the following method:</p> <ul style="list-style-type: none">• Each thread exclusively accesses these objects.
2	<p>During unmarshal and marshal, processing similar to the schema generator might be executed within the product. Therefore, you must also see the notes on the schema generator.</p>
3	<p>Do not execute unmarshal or marshal in the verification mode for the schema document using <code>xs:ENTITY</code> type or <code>xs:ENTITIES</code> type. If you execute unmarshal or marshal, an error indicating 'Entity is not declared' occurs.</p> <p>(Example)</p> <p>KECX06252-E UndeclaredEntity: Entity 'idl' is not declared.</p>
4	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ul style="list-style-type: none">• The <code>unmarshal(org.w3c.dom.Node node)</code> method of <code>Unmarshaller</code> is executed.• A <code>Node</code> object other than <code>org.w3c.dom.Document</code> or <code>org.w3c.dom.Element</code> is specified in the <code>node</code> parameter. <p>Operations of Cosminexus XML Processor</p> <p>An exception <code>IllegalArgumentException</code> occurs.</p> <p>(Example)</p> <p><code>java.lang.IllegalArgumentException: Unexpected node type: [#text: String]</code></p>
5	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ul style="list-style-type: none">• An aligned event callback method (<code>beforeUnmarshal/afterUnmarshal</code>) that throws the exception is defined in the class.• Marshal is executed in such a way so that the instance document invoked by the above event callback method is output. <p>Operations of Cosminexus XML Processor</p> <p>The marshal process will terminate, but an exception <code>IllegalStateException</code> will be thrown.</p>
6	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none">1. <code>@XmlJavaTypeAdapter</code> is specified at the package level.2. <code>@XmlJavaTypeAdapter</code> is specified for the class or enumeration type.3. The instance document contains a data type definition in which the <code>@XmlJavaTypeAdapter</code> annotation specified in condition 1. and 2. is enabled.4. Unmarshal or marshal is executed based on the conditions 1. to 3. <p>Operations of Cosminexus XML Processor</p> <p>The <code>@XmlJavaTypeAdapter</code> annotation specified in the package level of the class is prioritized than the <code>@XmlJavaTypeAdapter</code> annotation for the class or enumeration type. As the operations differ from the standard specifications, do not perform process for such a java source.</p>
7	<p>[Conditions]</p> <p>When the Java contents tree is changed within the <code>handleEvent</code> method.</p> <p>Operations of Cosminexus XML Processor</p> <p>An exception <code>MarshalException</code> is thrown.</p>
8	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none">1. Marshal is executed.

No.	Notes
	<p>2. An argument <code>jaxbElement</code> of the <code>marshal</code> method is <code>JAXBElement</code> object.</p> <p>3. Either <code>localPart</code> or <code>prefix</code> of the <code>QName</code> object specified in the <code>name</code> parameter of the <code>JAXBElement</code> constructor specified in condition 2. is not <code>NCName</code>.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>marshal</code> method will terminate normally, but an XML document that is not in a proper format is generated.</p>
9	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The value element of <code>@XmlEnum</code> is <code>javax.xml.namespace.QName.class</code>. 2. The value element of <code>@XmlEnumValue</code> is a literal character string indicating <code>QName</code>. 3. A Java class that includes the condition 1. and 2. is specified and executed in the argument of the <code>JAXBContext.newInstance()</code> method. <p>Operations of Cosminexus XML Processor</p> <p>An exception <code>JAXBException</code> occurs. Do not perform processing for such a java source.</p>
10	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ul style="list-style-type: none"> • The <code>getURL</code> method of the <code>ValidationEventLocator</code> interface is executed. • The URL of the object for which <code>marshal</code> and <code>unmarshal</code> is to be performed includes characters that cannot be specified in RFC2396 (also includes the cases when the URL is generated automatically from the directory name or file name). <p>Operations of Cosminexus XML Processor</p> <p>The return value of the <code>getURL</code> method includes characters that cannot be specified in RFC2396 as it is.</p>
11	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. <code>@XmlAttribute</code> is specified in the static field whose type is <code>primitive type</code>. 2. <code>Marshal</code> is performed for the Java source specified in condition 1. <p>Operations of Cosminexus XML Processor</p> <p>An invalid exception occurs in <code>marshal</code>. Such a java source is out of scope for processing.</p> <p>(Example)</p> <p>Exception in thread "main" <code>java.lang.IncompatibleClassChangeError</code></p>
12	<p>[Conditions]</p> <p>When <code>null</code> is specified in an argument of the <code>setter</code> method of a Java value class.</p> <p>Operations of Cosminexus XML Processor</p> <p>The <code>NullPointerException</code> exception occurs.</p> <p>[Corrective action]</p> <p>Modify the code of the <code>setter</code> method invocation part as follows:</p> <pre>(Before modification) items.setId(null); (After modification) items.setId(new String[] {});</pre>
13	<p>[Conditions]</p> <p>When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The <code>xsi:type</code> attribute is specified in an element of the instance document. 2. The type specified in condition 1. is not the JAXB map type. 3. <code>Unmarshal</code> is executed in condition 1. and 2. <p>Operations of Cosminexus XML Processor</p> <p><code>Unmarshal</code> terminates normally.</p> <p>The element will be unmarshalled without substitution of the type by <code>xsi:type</code>.</p>
14	<p>When an error occurs due to the execution of <code>marshal</code> in the verification mode, an invalid XML (XML that is not in a proper format) might be output for the contents up to the occurrence of the error.</p>

No.	Notes
15	When marshal is executed by specifying the <code>jaxb.formatted.output</code> property, the linefeed code of the output XML document will be 0x0A.
16	When marshal is executed, the <code>standalone="yes"</code> pseudo attribute will be output to the XML declaration of the output XML document.
17	<p>[Conditions] When a Node object other than Element, Document, or DocumentFragment is specified in the argument node of the marshal(Object jaxbElement, Node node) method of Marshaller interface.</p> <p>Operations of Cosminexus XML Processor An exception DOMException will be thrown.</p> <p>(Example) <pre>org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: KECX01304-E An attempt was made to insert a node where it is not permitted.</pre></p>
18	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. An element of <i>xs:list</i> type for which the default value is specified is declared in the schema document. 2. Condition 1. is applied to the schema compiler, and the XML document is unmarshalled using the generated Java value class. 3. The instances of the Java value class specified in condition 2. are acquired by the get method. <p>Operations of Cosminexus XML Processor The default values defined in the schema document are not included in the acquired instances.</p>
19	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. @XmlAttachmentRef is added to any one of the field, JavaBean property, or parameter. 2. The Java source specified in condition 1. is input to the schema generator. 3. Validation is performed in the schema document generated in condition 2. <p>Operations of Cosminexus XML Processor The value of the type attribute in the element definition or attribute definition of the generated schema will be {http://ws-i.org/profiles/basic/1.1/xsd}swaRef. The definition of the name space {http://ws-i.org/profiles/basic/1.1/xsd} is generated as <code>xs:import:</code> <pre><xs:import namespace="http://ws-i.org/profiles/basic/1.1/xsd" schemaLocation="http://ws-i.org/profiles/basic/1.1/swaref.xsd"/></pre> Therefore, when validation is performed in the generated schema document, the XML parser references <code>http://ws-i.org/profiles/basic/1.1/swaref.xsd</code> using the http protocol. By setting up the correct <code>org.w3c.dom.ls.LSResourceResolver</code> instance using the appropriate <code>javax.xml.validation.Validator#setResourceResolver</code> method, <code>swaref.xsd</code> can be referenced at any place of existence.</p>
20	<p>[Conditions] When all of the following conditions are applicable:</p> <ol style="list-style-type: none"> 1. The <code>jaxb:typesafeEnumClass</code> custom binding is specified. 2. The schema document specified in condition 1. is input into the schema compiler. 3. The <code>fromValue</code> method of the enumeration type is generated in condition 2., and is invoked by specifying value other than the enumeration constant in its argument. <p>Operations of Cosminexus XML Processor An exception <code>IllegalArgumentException</code> occurs.</p>
21	<p>[Conditions] When all of the following conditions are applicable:</p> <ul style="list-style-type: none"> • <code>package-info.java</code> exists. • After executing <code>JAXBContext.newInstance(Class... classesToBeBound)</code>, the schema is generated, and the methods of <code>unmarshal</code> and <code>marshal</code> are executed. <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	The package-info.java will not be processed if not compiled beforehand.
22	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Add <code>@XmlAnyElement(lax = true)</code> to either the <code>Object</code> type field or the <code>JavaBean</code> property. • An element is not in conformity with the schema element definition, but the type name specified in the <code>xsi:type</code> attribute of that element will unmarshal the instance document that is defined in the schema. <p>Operations of Cosminexus XML Processor The DOM node will be generated without generating a JAXB object (Java class object corresponding to the schema element definition).</p>
23	<p>[Conditions] When <code>null</code> is specified in the argument of the <code>setEventHandler</code> method in any of the following classes:</p> <ul style="list-style-type: none"> • <code>Unmarshaller</code> class • <code>Marshaller</code> class • <code>Binder<Node></code> class <p>Operations of Cosminexus XML Processor The event handler acquired using the <code>getEventHandler</code> method is the default event handler (<code>DefaultValidationEventHandler</code>) of JAXB1.0, and differs from the event handler acquired without executing the <code>setEventHandler</code> method. In such cases, the operations are not guaranteed.</p>
24	<p>[Conditions] When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • Using the <code>setProperty</code> method of the <code>Binder<Node></code> class, set up a value other than the default value in the <code>jaxb.encoding</code> property. • Execute the <code>marshal</code> method of the <code>Binder<Node></code> class. <p>Operations of Cosminexus XML Processor The specified properties are not valid.</p>
25	<p>Condition When all the following conditions are satisfied:</p> <ul style="list-style-type: none"> • The <code>jaxb.formatted.output</code> property is not set in the <code>setProperty</code> method of the <code>Binder<Node></code> class, or <code>false</code> is set up. • The <code>marshal</code> method of the <code>Binder<Node></code> class is executed. <p>Operations of Cosminexus XML Processor Regardless of the property settings, XML data with processed linefeeds and indents is output.</p>
26	<p>[Conditions] When the following is specified in <code>unmarshal(XmlNode xmlNode and Class<T> declaredType)</code> of the <code>javax.xml.bind.Binder<XmlNode></code> class:</p> <ul style="list-style-type: none"> • Specify <code>xmlNode</code> that is not in conformity with a class specified in <code>declaredType</code>. <p>Operations of Cosminexus XML Processor Exceptions do not occur. In such cases, operations for executing JAXB are not defined, so the operations are not guaranteed.</p>
27	<p>[Conditions] When the following is specified in <code>updateXML(Object jaxbObject and XmlNode xmlNode)</code> of the <code>javax.xml.bind.Binder<XmlNode></code> class:</p> <ul style="list-style-type: none"> • In the <code>jaxbObject</code> argument, specify a JAXB object that is not associated to <code>unmarshal</code> and <code>marshal</code>. • Specify the DOM node that was deleted after associating with the <code>xmlNode</code> argument. <p>Operations of Cosminexus XML Processor In such cases, operations for executing JAXB are not defined, so the operations are not guaranteed.</p>
28	<p>[Conditions] When the <code>updateXML</code> method of the <code>javax.xml.bind.Binder<XmlNode></code> class is executed.</p> <p>Operations of Cosminexus XML Processor</p>

No.	Notes
	<p>A new XML tree is created without updating the already existing XML tree that is associated to the JAXB object. Therefore, if the following information is included in the existing XML tree, this information will not be maintained after the method is executed:</p> <ul style="list-style-type: none"> • Comment • PI • XML elements or attributes that are not bound with JAXB <p>Also, a return value of <code>updateXML (Object jaxbObject and XmlNode xmlNode)</code> is not the node as that of the <code>xmlNode</code> argument. Therefore, when executing the <code>updateXML</code> method repetitively, do not specify the same value in the <code>xmlNode</code> argument.</p>
29	<p>[Conditions] When executing the <code>updateXML</code> method of the <code>javax.xml.bind.Binder<XmlNode></code> class.</p> <p>Operations of Cosminexus XML Processor Association of the JAXB object tree and an XML node having the implemented <code>javax.xml.bind.Binder<XmlNode></code> class is invalid. Therefore, after executing the method, the execution result of specifying the return value in the <code>getJAXBNode (XmlNode xmlNode)</code> argument is <code>null</code>. Also, when executing the method repetitively, the <code>NullPointerException</code> exception occurs.</p> <p>Workaround Perform the following changes, and in addition to invoking of the <code>updateXML</code> method, also invoke the <code>updateJAXB</code> method of the <code>javax.xml.bind.Binder<XmlNode></code> class.</p> <p>Before change: <code>binder.updateXML (rootbinder) ;</code> After change: <code>Node node = binder.updateXML (rootbinder) ;</code> <code>rootbinder = (Root)binder.updateJAXB (node) ;</code></p>
30	<p>[Conditions] When any of the following condition is satisfied:</p> <ul style="list-style-type: none"> • During the unmarshalling, an error occurs when a character string of the XML data is converted into a value of the target Java data type. • During the marshalling, an error occurs when the data of the Java contents tree is converted into the text expression of the Java contents tree. <p>Operations of Cosminexus XML Processor Events (<code>parseConversionEvent</code> for unmarshalling or <code>printConversionEvent</code> for marshalling) are not generated.</p>
31	<p>[Conditions] When all the following conditions are satisfied:</p> <ol style="list-style-type: none"> 1. Specify the <code>xsi:type</code> attribute in an XML document. 2. In the attribute value of step 1, specify a namespace prefix that is not used in the element and attribute names. 3. Unmarshal the XML document of step 1. 4. Marshal the object that is unmarshal in step 3 in the verification mode. <p>Example: <code><foo xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"></code> <code><a/></code> <code><b xsi:type="xsd:string" xmlns:xsd="http://www.w3.org/2001/XMLSchema">abc</code> <code><c xsi:type="xsd:int" xmlns:xsd="http://www.w3.org/2001/XMLSchema">123</c></code> <code></foo></code></p> <p>Operations of Cosminexus XML Processor The following error is reported: KECX06253-E UndeclaredPrefix: Cannot resolve 'xsd:string' as a QName: the prefix 'xsd' is not declared. Also, if executing the marshalling without the verification mode, an XML document, in which the required name space is not declared, will be output.</p>
32	<p>[Conditions] When all the following conditions are satisfied:</p>

No.	Notes
	<ol style="list-style-type: none"> 1. The field A is declared as the <code>java.util.Calendar</code> class. 2. Annotation is performed done for the field of step 1 using <code>@XmlSchemaType (name="date")</code>. 3. Marshal a class with the fields that satisfy step in 1 and 2. <p>Operations of Cosminexus XML Processor</p> <p>The contents of the elements within the XML document that corresponds to the field A is output in the <code>xs:dateTime</code> type and not in the <code>xs:date</code> type.</p> <p>Therefore, if you use the schema the schema that is generated with the <code>generateSchema</code> method of the <code>JAXBContext</code> class to perform the schema verification, a verification error will occur.</p> <p>Workaround</p> <p>To output in the <code>xs:date</code> format while marshalling, use the <code>javax.xml.datatype.XMLGregorianCalendar</code> class and not the <code>java.util.Calendar</code> class.</p>
33	<p>[Conditions]</p> <p>When executing the <code>getOffset</code> method of the <code>ValidationEventLocator</code> interface.</p> <p>Operations of Cosminexus XML Processor</p> <p>The return value of the <code>getOffset</code> method is always <code>-1</code>.</p>

6.21 General XML Processor related notes

This section describes other notes related to general usage of the XML Processor.

Table 6–47: Notes related to general usage of the XML Processor

No.	Notes
1	When an XML Schema is used for validation processing, exceptions or errors that are not defined in the API specifications might occur due to some factors. With Java programs that use an XML Schema for validation processing, implement exception processing that can catch <code>java.lang.Throwable</code> and take appropriate action, on condition that such exceptions or errors are thrown.
2	<p>The JAR file provided by XML Processor contains packages whose names begin with the strings listed in 5.2 Package Names Used by Cosminexus XML Processor. If there is a user application that has an implementation that includes the same packages that are provided by XML Processor, the classes of the packages provided by XML Processor are loaded preferentially.</p> <p>However, do not include such an implementation in the user application if the <code>Sealed</code> attribute is set to <code>true</code> in the manifest file for the JAR file that contains those packages. If you do so, a <code>SecurityException</code> occurs.</p> <p>For details about package names, see 5.2 Package Names Used by Cosminexus XML Processor.</p>

Appendixes

A. Differences Between Versions

This appendix describes the behaviors that differ from the previous version of Cosminexus XML Processor.

A.1 Differences in Behaviors of XSLT Between Versions

The following table lists and describes the differences in behaviors of XSLT between versions.

Table A–1: Differences in XSLT operations (Comparing version 08-50 and version 08-70)

No.	Condition	Cosminexus XML Processor operations	
		In version 08-50	In version 08-70 and later versions
1	If <code>XMLEventReader</code> acquired by using the <code>XMLInputFactory.createXMLEventReader()</code> method is specified in the <code>StAXSource</code> constructor argument, and is used as an input source for the <code>Transformer.transform()</code> method	The processing command outside the root element of the input XML document is ignored.	The processing command outside the root element of the input XML document is also transformed.
2	When both of the following conditions are satisfied: <ul style="list-style-type: none">• The elements in the input XML document for transformers belong to a non-XSLT namespace• Transformation is performed by using <code>StAXResult</code> created from <code>XMLEventWriter</code>	An error occurs during transformation or the transformation results are invalid.	Processed normally.
3	When both the following conditions are satisfied: <ul style="list-style-type: none">• Transformation is performed by using <code>xsl:sort</code> in the style sheet of the transformer• Japanese language is specified in the sort key of the style sheet of the transformer	The output results of sorting are different in J2SE 5.0 and Java SE 6.	In version 08-70 or earlier versions Output results of sorting are different in J2SE 5.0 and Java SE 6. In version 09-00 and later versions There is no difference as only Java SE 6 is used.
4	When the output method is <code>xml</code> , and the characters that cannot be expressed with the specified output encoding are output	The output results might differ in J2SE 5.0 and Java SE 6.	In versions up to 08-70 The output results might differ in J2SE 5.0 and Java SE 6. In version 09-00 or later There are no differences because only Java SE 6 is used.

A.2 Differences in the JAXB operation

The following table lists and describes the status-wise differences in JAXB operations.

Table A–2: Differences in JAXB operations (csmxjc command)

No.	Condition	Cosminexus XML Processor operations	
		In version 08-50	In version 08-70 and later versions
1	If the <code>typesafeEnumClass</code> custom binding exists in a simple XML schema document in which the <code>enumeration</code> facet does not exist	The <code>csmxjc</code> command terminates normally.	An error occurs indicating that the specification of the <code>typesafeEnumClass</code> custom binding is invalid.
2	When one of the following is satisfied: <ul style="list-style-type: none"> If <code>generateElementClass="true"</code> or <code>generateElementClass="1"</code> is specified in the <code>globalBindings</code> custom binding If <code>class</code> custom binding is specified in the <code>element</code> element of the XML schema document 	The reference property of the element class is annotated by using a data type of the element (<code>@XmlElement</code> or <code>@XmlElements</code>). Note that the default constructor is not output to the source corresponding to that element.	The reference property of the element class is annotated by using a class type of the element (<code>@XmlElementRef</code> or <code>@XmlElementRefs</code>). The default constructor is output.
3	If the class custom binding is specified for the element element in a group element and that group element is referenced from other multiple types	Among the element elements in the group element, the factory method that creates the objects of the elements not specifying the class custom binding is not generated.	Among the element elements in the group element, the factory method that creates the objects of the elements not specifying the class custom binding is generated.
4	If <code>collectionType="indexed"</code> is specified for <code>globalBindings</code> or <code>property</code> custom binding	The type for the element element of the source generated by the <code>csmxjc</code> command is <code>java.util.List</code> .	The type for the element element of the source generated by the <code>csmxjc</code> command is an array.
5	If the <code>csmxjc</code> command is executed when the <code>factoryMethod</code> custom binding is specified in the type definition or the anonymous type element definition	An error occurs.	Terminates normally.
6	If the source generated by the <code>csmxjc</code> command includes the <code>@XmlElementRef</code> annotation type [#]	The required element is not output.	The required element is output.
7	If the name element of the <code>@XmlAttribute</code> annotation type is the same as the field name to be generated	The name element is not output.	The name element is output.

The required element of the `@XmlElementRef` annotation type is now supported in JAXB 2.2.

Table A–3: Differences in JAXB operations (csmschemagen command)

No.	Condition	Cosminexus XML Processor operations	
		In version 08-50	In version 08-70 and later versions
1	If <code>@XmlAccessorType</code> or <code>@XmlAccessorType(XmlAccessorType.UNDEFINED)</code> is specified in the Java source	The XML schema document created by the <code>csmschemagen</code> command becomes the <code>sequence</code> element.	The XML schema document created by the <code>csmschemagen</code> command becomes the <code>all</code> element.
2	If <code>@XmlElementDecl</code> is specified in the Java source	The <code>nillable="true"</code> attribute is output to the element element of the global definition for the XML schema document generated by the <code>csmschemagen</code> command.	The <code>nillable="true"</code> attribute is not output to the element element of the global definition for the XML schema document generated by the <code>csmschemagen</code> command.

Table A–4: Differences in JAXB operations (Runtime)

No.	Condition	Cosminexus XML Processor operations	
		In version 08-50	In version 08-70 and later versions
1	If a class containing variables of the <code>java.lang.Object</code> type annotated by the <code>@XmlAttribute</code> or <code>@XmlValue</code> annotation types, or a package name containing that class, is specified in the <code>newInstance</code> method for the <code>JAXBContext</code> class	An <code>IllegalArgumentException</code> is thrown.	A <code>NullPointerException</code> is thrown.
2	If the <code>package-info</code> class exists in the package specified in the <code>newInstance</code> method for the <code>JAXBContext</code> class, and if the namespace is specified by using the <code>XmlNs</code> annotation type	The namespace declaration is not output to the XML document that is output with marshal.	The namespace declaration is output to the XML document that is output with marshal.
3	If an XML document that only contains blank text in the JavaBean property annotated by the <code>@XmlMixed</code> annotation type, is un-marshaled	The text object is not generated in the object generated by un-marshal.	A blank text object is generated.
4	In a schema validation or event handler, if an <code>Unmarshaller</code> that does not check errors is used to un-marshal a character string for a JavaBean property with a primitive type numeric value	An <code>IllegalAccessError</code> occurs.	Un-marshal processing terminates normally.
5	If the <code>beforeUnmarshal</code> method or <code>afterUnmarshal</code> method is implemented for the classes mapped to JAXB and if that method throws an exception during un-marshal	Un-marshal processing is interrupted.	An <code>UnmarshallerException</code> is thrown and un-marshal processing is interrupted.
6	If null is specified in the first parameter of the <code>newInstance(Class[], Map<String, ?>)</code> method and <code>newInstance(Class[])</code> method for the <code>JAXBContext</code> class	A <code>NullPointerException</code> is thrown.	An <code>IllegalArgumentException</code> is thrown.
7	If a value other than 0, 1, false, and true is specified in the <code>parseBoolean</code> method for the <code>DatatypeConverter</code> class	false is returned.	An <code>IllegalArgumentException</code> is thrown.
8	If you specify only a space in the <code>parseDecimal</code> method for the <code>DatatypeConverter</code> class	<code>java.lang.NumberFormatException</code> is thrown.	An exception is thrown.
9	If the value to be un-marshaled in the enum class contains a linefeed or a space	The processing result is invalid.	The value is un-marshaled correctly.

A.3 Differences in the operation of the `javax.xml.datatype` package

The following table describes the differences in the operation of the `javax.xml.datatype` package:

Table A–5: Differences in the operation of the javax.xml.datatype package (Comparison of cases when JAXP1.3 is used on J2SE 5.0 and JAXP1.4 is used on Java SE 6)

No.	Condition	Execution result	
		In J2SE 5.0	In Java SE 6
1	Content of the <code>Duration</code> object returned by the <code>newDurationYearMonth(long)</code> method of the <code>javax.xml.datatype.DatatypeFactory</code> class	Data other than year and month is also maintained.	Data other than year and month is not maintained.
2	When <code>null</code> is specified in the parameter of the <code>equals</code> method of the <code>javax.xml.datatype.Duration</code> class	<code>NullPointerException</code> occurs.	<code>false</code> is returned. <code>NullPointerException</code> does not occur.

A.4 Differences in the operation of the high-speed parse functionality

The following table describes the differences in the operation of the high-speed parse functionality:

Table A–6: Differences in the operation of the high-speed parse functionality (comparison of cases when JAXP1.3 is used on J2SE 5.0 and JAXP1.4 is used on Java SE 6)

No.	Condition	Execution result	
		In J2SE 5.0	In Java SE 6
1	When a <code>File</code> object including a character (such as a Japanese character or a blank) that is not allowed as the XML document name by RFC2396 is specified in the following methods: <ul style="list-style-type: none"> <code>parse(File)</code> method of the <code>DocumentBuilder</code> class <code>parse(File, DefaultHandler)</code> method of the <code>SAXParser</code> class 	The following file names will be as specified in the <code>File</code> object: <ul style="list-style-type: none"> XML document name of the tuning summary file that is to be analyzed File name of the tuning details file XML document name of the tuning details file that is to be analyzed 	The following file names are encoded with <code>%xx</code> : <ul style="list-style-type: none"> XML document name of the tuning summary file that is to be analyzed File name of the tuning details file XML document name of the tuning details file that is to be analyzed

A.5 Differences in the operation of StAX

The following table describes the differences in the operation of StAX:

Table A–7: Differences in the operation of StAX (comparison of cases when JAXP1.3 is used on J2SE 5.0 and JAXP1.4 is used on Java SE 6)

No.	Condition	Execution result	
		In J2SE 5.0	In Java SE 6
1	When the <code>getCause</code> method is executed in the object generated in the <code>javax.xml.stream.XMLStreamException(Throwable)</code> constructor	<code>null</code> is returned.	The <code>Throwable</code> object specified in the constructor is returned.

A.6 Differences in the operations of the Apache-specific functionality

Depending on the Apache-specific functionality supported in 08-70, the operations of the XML parser and XSLT/XSLTC might differ from the operations in versions 08-50 or earlier. For details on the Apache-specific functionality, see [3.5 Apache-specific functions](#).

A.7 Differences in Cosminexus 09-80 or later operations

The following table shows the Cosminexus XML Processor operations that change when it operates in Cosminexus 09-80 and later versions.

Table A–8: Differences in the operation of Cosminexus XML Processor

No.	Condition	Cosminexus XML Processor operations	
		Cosminexus 09-70 and earlier versions	Cosminexus 09-80 and later versions
1	In an XML document encoded in UTF-16 big-endian, ISO-10646-UCS-4 is specified for the encoding attribute of the XML declaration.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01021-E error is reported.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01024-E error is reported.
2	In an XML document encoded in UTF-16 little-endian, ISO-10646-UCS-4 is specified for the encoding attribute of the XML declaration.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01024-E error is reported.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01021-E error is reported.
3	In an XML document encoded in ISO-10646-UCS-2 big-endian, ISO-10646-UCS-4 is specified for the encoding attribute of the XML declaration.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01021-E error is reported.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01024-E error is reported.
4	In an XML document encoded in ISO-10646-UCS-2 little-endian, ISO-10646-UCS-4 is specified for the encoding attribute of the XML declaration.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01024-E error is reported.	When the XML document is parsed by using a DOM parser or SAX parser, a KECX01021-E error is reported.

B. Support Range of JAXB Specifications

This appendix describes the support range of JAXB specifications. For details about the functions, see the manual of each product.

B.1 Support range of the JAXB functions

The following table describes the support range of the JAXB functions:

Table B–1: Support range of JAXB functions

JAXB function			Overview	Supported or not	
Schema compiler	Default binding		Performs default conversion.	Y	
	Custom binding		Customizes the conversion on the basis of binding declaration.	Y	
		Schema with inline annotation	Describes the binding declaration in the schema document.	Y	
		External binding declaration	Describes the binding declaration in the external file.	Y	
		Extending the binding language	Defines the vendor-specific extended binding declaration.	N	
Schema generator	Default mapping		Performs default conversion.	Y	
	Custom mapping		Customizes the conversion on the basis of the Java annotations.	Y	
Binding framework	Marshal		Outputs the Java objects as XML documents.	Y	
		Event callback		This is the callback method of the event generated during marshalling.	Y
			Class-defined callback method	This is the callback method within the class mapped by JAXB invoked before and after marshalling.	Y
			External listener	This is an external callback method invoked by the event that occurs during marshalling.	Y
		Marshalling properties		Various properties for controlling marshalling.	Y
		Validation		Performs validation during marshalling.	Y
		Unmarshal		Reads XML document as a Java object.	Y
		Event callback		This is the callback method of the event generated during un-marshalling.	Y
			Class-defined callback method	This is the callback method that exists within the class mapped by JAXB invoked before and after un-marshalling.	Y
			External listener	This is an external callback method invoked by the event that occurs during un-marshalling.	Y
		Validation		Performs validation during un-marshalling.	Y
	Introspector		This is a function for accessing information about mapping of the Java object that is mapped by JAXB.	Y	

JAXB function		Overview	Supported or not
	Binder	This is a function for achieving synchronization of the Java object mapped by JAXB, and the XML information set.	Y

Legend:

Y: All functions are supported.

N: Not supported

B.2 Support range for JAXB characters

You can specify any character string in the schema document that is the input for schema compiler and in the Java class that is the input for schema generator. [Table B-2](#) describes the support range of the characters of java source input into the schema generator handled in JAXB, and [Table B-3](#) describes the support range of characters of the schema document input into the schema compiler.

Table B–2: Support range of characters of java source input into the schema generator

No.	Specification location of any character string	Support range of the character
1	Class name Method name Field name	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> • Must comprise of alphanumeric characters and underlines. • Must conform to the syntax rules for the NCName of XML.
2	Enumeration constant	The following condition must be fulfilled: <ul style="list-style-type: none"> • Must comprise of alphanumeric characters and underlines.
3	prefix element of @XmlNs	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> • Must comprise of alphanumeric characters and underlines. • Must conform to the syntax rules for the NCName of XML.
4	Elements of @XmlEnumValue defaultValue element of @XmlElement or @XmlElementDecl	The following condition must be fulfilled: <ul style="list-style-type: none"> • Must comprise of Unicode Basic Latin and Japanese^{#1}.
5	name element of the JAXB mapping annotation	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> • Must comprise of alphanumeric characters, underlines, and Japanese^{#1}. • Must conform to the syntax rules for the NCName of XML.
6	namespace element of the JAXB mapping annotation	The following condition must be fulfilled: <ul style="list-style-type: none"> • Must be an URI character string provided in RFC 2396.^{#2}
7	factoryMethod element of @XmlType	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> • Must comprise of alphanumeric characters and underlines. • Must conform to the syntax rules for the Java identifier.
8	value element of @XmlMimeType	The following condition must be fulfilled: <ul style="list-style-type: none"> • Must be a MIME-type text expression.

#1

Characters included in the Unicode Hiragana, Katakana, and CJK Unified Ideographs category.

#2

RFC2732 is not supported (IPv6 not supported).

Table B–3: Support range of characters of the schema document input into the schema compiler

No.	Specification location of any character string	Support range of character
1	Attributes of the schema element that specifies the <code>xs:anyURI</code> type (such as the <code>targetNamespace</code> attribute of the <code>xs:schema</code> element)	The following condition must be fulfilled: <ul style="list-style-type: none"> Must be an URI character string provided in RFC 2396.^{#1}
2	Attributes of the schema element that specifies the <code>xs:NCName</code> type (such as the <code>name</code> attribute of the <code>xs:element</code> element)	All of the following conditions must be fulfilled when the output name is not changed with the custom binding declaration: <ul style="list-style-type: none"> Must comprise of alphanumeric characters and underlines. Must conform to the syntax rules for the <code>NCName</code> of XML.
		All of the following conditions must be fulfilled when the output name is changed with the custom binding declaration: <ul style="list-style-type: none"> Must comprise of alphanumeric characters, underlines, and Japanese^{#2}. Must conform to the syntax rules for the <code>NCName</code> of XML.
3	Attributes of the schema element that specifies the <code>xs:QName</code> type (such as the <code>type</code> attribute of the <code>xs:element</code> element)	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> Must be a character string of the <code>QName</code> format. The local name must be within the range of item no. 2. The prefix must comprise of alphanumeric characters and underlines.
4	Attribute of the schema element that specifies the <code>xs:string</code> type (such as the <code>fixed</code> or <code>default</code> attribute of the <code>xs:element</code> element, or the <code>fixed</code> or <code>default</code> attribute of the <code>xs:attribute</code> element) value attribute of the <code>jaxb:typesafeEnumMember</code> element	The following condition must be fulfilled: <ul style="list-style-type: none"> Must comprise of Unicode Basic Latin and Japanese^{#2}.
5	value attribute of the <code>xs:enumeration</code> element	All of the following conditions must be fulfilled when the enumeration constant is not changed with the custom binding declaration: <ul style="list-style-type: none"> Must comprise of alphanumeric characters and underlines. Must conform to the syntax rules of the Java enumeration constant.
		All of the following conditions must be fulfilled when the enumeration constant is changed with the custom binding declaration: <ul style="list-style-type: none"> Must comprise of Unicode Basic Latin and Japanese^{#2}.
6	name attribute of the JAXB element suffix attribute of the JAXB element prefix attribute of the JAXB element parseMethod or printMethod attribute of the <code>jaxb:javaType</code> element	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> Must comprise of alphanumeric characters and underlines. Must conform to the syntax rules for the Java identifier.
7	name attribute of the <code>jaxb:package</code> element ref attribute of the JAXB element collectionType attribute of the JAXB element implClass attribute of the <code>jaxb:class</code> element	All of the following conditions must be fulfilled: <ul style="list-style-type: none"> Must comprise of alphanumeric characters, underlines, and periods (.). Must conform to the syntax rules for the Java package name.
8	Element contents of the <code>jaxb:javadoc</code> element	All of the following conditions must be satisfied: <ul style="list-style-type: none"> Must be configured of Basic Latin of Unicode and Japanese language^{#2}. Must conform to the syntax rules of the Java command.

#1

RFC2732 is not supported (IPv6 not supported).

#2

Characters included in the category of Unicode Hiragana, Katakana, and CJK Unified Ideographs

B.3 Support range of functions assumed as vendor-specific according to JAXB specifications

The following table describes the support range of functions assumed as vendor-specific according to the JAXB specifications:

Table B–4: Support range of functions assumed as vendor-specific according to JAXB specifications

Corresponding location in JSR 222	Page number	Content of the vendor-specific function	Function support in Cosminexus XML Processor
4.3 General Validation Processing	35	Unmarshal-time validation (Validation during unmarshal)	The functions for validation is supported during unmarshal.
		On-demand validation (On-demand validation)	The on-demand validation function is not supported.
		Fail-first validation (Fail-first validation)	The fail-first validation function is not supported.
4.5 Marshalling	41	Potentially properties (Additional properties for marshal)	There are no <i>Properties for marshaling</i> that are specific to Cosminexus XML Processor.
5.2 Java Package	50	Property setter/getter (Additional properties of the Java value class)	There are no <i>Properties of the Java value class</i> that are specific to Cosminexus XML Processor.
5.5.1 Simple Property	57	TypeConstraint validation (Type constraint validation)	The type constraint validation function is not supported.
6.13 Modifying Schema-Derived Code	153	Distinguish between generated and user added code (Identification of the auto-generated Java sources)	The identification of auto-generated Java sources is supported by <code>-mark-generated</code> of the <code>csmxjc</code> command.
7.1.1 Extending the Binding Language	159	Extending the Binding Language (Extension of the binding declaration)	The binding declaration extension function is not supported.
7.5<globalBindings> Declaration 7.8<property> Declaration	167 189	enableFailFastCheck (enableFailFastCheck attribute)	The <code>enableFailFastCheck</code> attribute is not supported.
Compatibility	294	non-default operating modes for binding schema languages (Binding of other schema languages)	The schema compiler does not support schema languages other than the XML Schema.
Compatibility	294	non-default operating modes for mapping Java types to schema languages (Mapping to other schema languages)	The schema generator does not support schema languages other than the XML Schema.
Appendix G	355	Validator for JAXB 1.0 schema-derived classes	In Cosminexus XML Processor, you cannot handle classes created in JAXB 1.0.

Corresponding location in JSR 222	Page number	Content of the vendor-specific function	Function support in Cosminexus XML Processor
		(JAXB1.0 compatibility specification)	

C. Efficiently using XML Processor

This appendix describes the features and notes on the functionality that helps you to use XML Processor efficiently.

The following table describes the classification of the XML Processor functionality.

Table C–1: Classification of the XML Processor functionality

Classification of the functionality			Main Java packages related to the functionality	Overview of the functionality
JAXP	Basic functionality	DOM parser	<code>javax.xml.parsers</code> <code>org.w3c.dom</code>	XML parsing
		SAX parser	<code>javax.xml.parsers</code> <code>org.xml.sax</code>	
		StAX parser	<code>javax.xml.stream</code>	
	Application functionality	XSLT	<code>javax.xml.transform</code>	XML document conversion and output
		XPath	<code>javax.xml.xpath</code>	Selecting parts and acquiring information using the XPath expression
		Validation	<code>javax.xml.validation</code>	Schema validation
JAXB		JAXB	<code>javax.xml.bind</code>	Data exchange

C.1 Basic functionality of XML Processor

The basic functionality of XML Processor implements the XML document parsing that is required for the processing of the XML document. This subsection describes three types of parsing functionality (parser) provided by the basic functionality of XML Processor and the features of each parser.

(1) DOM parser

(a) Overview

The DOM parser parses all the elements, attributes, and text included in an XML document, and expands their contents as an object tree called the *DOM tree* in the memory. By using the DOM API, you can freely traverse the DOM tree and you can reference, update, or delete any data item.

(b) Optimum usage

- To process an XML document extensively
- To process XML document content that requires processing in a complicated order
- To edit the contents of an XML document

(2) SAX parser

(a) Overview

The SAX parser is an event-driven parser that parses an XML document sequentially from the beginning, and invokes predefined processing (callback method) whenever parsing events such as starting or stopping of elements, text, or processing instructions, occur.

When parsing starts, a callback method is invoked every time according to the type of the parsing event, and the parsing data, such as the element name or text data, is passed as an argument to the callback method.

You can perform processing in accordance with the content of the XML document by defining the processing content of the callback method in advance.

(b) Optimum usage

- To process an XML document with a simple structure
- For read-only processing, when the XML document contents need not be edited

(3) StAX parser

(a) Overview

The StAX parser, like the SAX parser, is an event-driven parser that parses an XML document sequentially from the beginning.

In the case of a StAX parser, parsing advances when the user requests a parsing event to the parser, and therefore, if you use a StAX parser, you can easily move ahead and interrupt the parsing process at any time. Also, by requesting a parsing event to a StAX parser, and assigning the processing as per the type and content of the parsing data acquired by requesting the parsing event, you can execute the processing in accordance with the contents of the XML document.

(b) Optimum usage

- To process the contents of a huge XML document partially

(4) Merits and demerits of each parser

The following table describes the merits and demerits of each parser.

Table C-2: Features of each parser

Parser type	Merit	Demerit
DOM parser	You can perform editing operations, such as addition or deletion, for an XML document.	Compared to the other parsers, the amount of memory consumed is large and the processing speed is slow.
SAX parser	Compared to the other parsers, the amount of memory consumed is small, and the processing speed is high.	You cannot perform editing operations, such as addition or deletion, for an XML document.
StAX parser	In addition to the above merits of the SAX parser, the user can control the execution or interruption of parsing.	The XML schema-based validation cannot be executed with the StAX parser alone.

C.2 Application functionality of XML Processor

The application functionality of XML Processor increases the convenience and maintainability by making the advanced and complex XML-related technology easily available.

The application functionality is implemented by using and integrating the basic functionality. Therefore, if the processing is simple, you can use the basic functionality instead of the application functionality.

For example, for a simple conversion processing such as renaming a specific element name in an XML document, you can implement the processing with the basic functionality of a parser such as a DOM parser, even if you do not use XSLT.

As a general tendency, the basic functionality has the advantage of a higher processing speed, but on the other hand, has the demerit that the basic functionality requires coding dependent on the XML document structure and that the coded code might be complicated. You must comprehensively determine which functionality is most appropriate for use based on the execution performance, memory performance, maintainability of the programs, and the features of the system and the data to be handled.

This subsection describes the application functionality of XML Processor.

(1) XSLT functionality

The XSLT functionality converts an XML document into an XML document with a different structure, or into an HTML document or plain text.

The conversion rules are defined in an XML document called *XSLT style sheet*, so you need not code complicated conversion processing as programs, and the development and maintenance of programs becomes easy.

(2) XPath functionality

The XPath functionality uses the expression (`XPath` expression) that indicates a specific position in an XML document to extract specific parts from an XML document and acquire information related to the document content and document structure.

You can use a simple code to check whether a specific element or attribute exists in the XML document and to acquire a part of the XML document.

(3) Validation functionality

The Validation functionality makes the XML schema-based validation processing more efficient by converting the syntax rules defined in the XML schema into objects.

To use one XML schema document for multiple validation processing, if you use the basic functionality, the XML schema document is parsed to ensure that a syntax rule is generated for every parsing. If you use the Validation functionality, the syntax rules of the XML schema can be re-used as objects, so the schema document is parsed only once.

(4) JAXB functionality

The JAXB functionality is used to easily code the data exchange processing that reads the data included in an XML document into a Java class, and outputs the data from the Java class as an XML document.

This functionality, when used together with additional functionality that auto-generates an XML schema document from a Java class and a Java class from the XML schema, offers great flexibility in changing the data structure as well.

C.3 Know-how for efficiently using XML Processor

This subsection provides the know-how for efficiently using XML Processor.

- The following methods exclusively control the threads internally. Therefore, if multiple threads invoke these methods concurrently, exclusive waiting might occur and the throughput might be affected:

1. The `newInstance` method of the factory classes of the following JAXP APIs:

```
javax.xml.datatype.DatatypeFactory
javax.xml.parsers.DocumentBuilderFactory
javax.xml.parsers.SAXParserFactory
javax.xml.transform.TransformerFactory
javax.xml.validation.SchemaFactory
javax.xml.xpath.XPathFactory
```

2. The `newDocumentBuilder` method of the `javax.xml.parsers.DocumentBuilder` class
3. The `transform` method of the `javax.xml.transform.Transformer` class
4. The `evaluate` method of the `javax.xml.xpath.XPath` class

Among these methods, the methods from 1 to 2 generate objects, so the processing is comparatively heavy. Therefore, re-use the generated objects and preferably do not invoke these methods frequently.

- If you need to create multiple `Transformers` from the same style sheet with the XSLT functionality, instead of creating `Transformer` directly from `TransformerFactory`, creating one `Templates` object, and then creating multiple `Transformers` from this object is advantageous with respect to the execution performance.
- To re-use the same XPath expression with the XPath functionality, using the `compile` method of the `XPath` class, and creating and re-using the `XPathExpression` object is advantageous with respect to the execution performance.
- You can use the DOM API to implement the same processing as the XPath functionality. Using the DOM API is advantageous in terms of the execution speed, but you must code a complicated process depending on the structure of the input XML document, and furthermore, when the structure of the XML document is changed, you must re-write the processing, so this process is inferior to XPath in terms of maintainability.
- Instead of the Validation functionality, you can also implement the validation processing using the XML schema with a DOM parser or SAX parser; however, when you repeat the validation processing by using the same schema document, the Validation functionality that can re-use the schema document parsing results as objects is advantageous with respect to the execution performance.
- The generation of the `JAXBContext` object increases the load, so re-using the objects as much as possible is advantageous with respect to the execution performance.

D. Glossary

Terminology used in this manual

For the terms used in the manual, see the *uCosminexus Application Server and BPM/ESB Platform Terminology Guide*.

Index

A

- application functionality [236](#)
- application functionality of XML Processor [236](#)
- attribute to use
 - type of the schema document to identify [107](#)

B

- basic functionality [234](#)
- basic functionality of XML Processor [234](#)

C

- case where XSLT does not report errors
 - note on XSLT [172](#)
- changes to program
 - XSLTC transformer function [147](#)
- class
 - high-speed parse support function [70](#)
 - XSLTC transformer [43](#)
- common note [199](#)
- compatibility option functionality [99](#)
- correspondence between the encoding specification for XML document and the applicable character encoding [40](#)
- Cosminexus XML Processor
 - difference between version [224](#)
 - JAXP function [20](#)
 - note [148](#)
 - overview [10](#)
 - package name [119](#)
 - positioning [14](#)
 - product feature [11](#)
 - range of features [15](#)
 - SAX2 features [102](#)
 - SAX2 properties [102](#)
- creating
 - Pre-Parse XML document [60](#)
- csm schemagen command (mapping Java to XML schema) [36](#)
- csmxjc command (binding XML schema to Java) [33](#)

D

- details
 - tuning details file [80](#)
 - tuning summary file [79](#)

- difference between version
 - differences in JAXB operation [224](#)
- difference in JAXB operation [224](#)
- differences in behaviors
 - XSLT [224](#)
- differences in Cosminexus 09-80 or later operations [228](#)
- differences in operation of high-speed parse functionality [227](#)
- differences in operation of javax.xml.datatype package [226](#)
- differences in operation of StAX [227](#)
- differences in operations of Apache-specific functionality [228](#)
- DOM parser [234](#)
 - general procedure for creating a sample program [121](#)
 - how to use Shift_JIS switch function [109](#)
 - note [153](#)
 - sample program [121, 137](#)

E

- efficiently using XML Processor [234](#)
- encoding
 - processable [17](#)
- error message in XML schema
 - W3C specification [86](#)

F

- feature (Apache-specific) [84](#)
- features and properties
 - how to use [101](#)
- flow
 - converting XML document in JAXP [12](#)
 - parsing XML document in JAXP [11](#)
- flow of processing
 - application for DOM parse [22](#)
 - application for evaluating XPath expression [26](#)
 - application for handling SAX event [30](#)
 - application for loading and saving XML document [29](#)
 - application for manipulating DOM tree [27](#)
 - application for processing date and time format data [27](#)
 - application for SAX parse [22](#)
 - application for validating XML document by using class included in javax.xml.validation package [25](#)

- application for XSLT-based transformation 23
- function
 - XML parser and XSLT transformer 15
- functionality for validated parsing compatibility 99

G

- general procedure for creating sample program
 - SAX parser 126
- general XML Processor related notes 222
- generating
 - preparsed object 68

H

- high-speed parse support function 57
 - class 70
 - coding example 75
 - flow of operation 59
 - how to output tuning information 77
 - how to use tuning information 77
 - output destination 78
 - overview 57
 - parse method 69
 - system property 78
 - tuning Pre-Parse XML document 76
 - type of tuning information 76
- how to create
 - program 117
- how to set schema document
 - XML document 107
- how to set the XML schema properties
 - DOM parser 106
- how to set up preparsed object
 - DocumentBuilder 112
 - SAXParser 113
 - XMLReader 113
- how to set XML schema properties
 - SAX parser 106
- how to use
 - features and properties 101
 - SAX2 features 102
 - SAX2 features and properties 102
 - SAX2 properties 103
 - XML schema property 106
 - XSLTC transformer 44
- how to use feature and property
 - XSLT 105

- how to use features and properties
 - high-speed parse support function 112
 - Shift_JIS switch function 109
- how to use property
 - StAX 104
- how to use Shift_JIS switch function
 - DOM parser 109
 - SAX parser 110
 - XSLT transformer 111

J

- javax.xml.datatype package 26
- javax.xml.namespace package 26
- javax.xml.parsers package 21
- javax.xml.stream.events package 23
- javax.xml.stream.util package 23
- javax.xml.stream package 22
- javax.xml.transform.dom package 24
- javax.xml.transform.sax package 24
- javax.xml.transform.stax package 24
- javax.xml.transform.stream package 24
- javax.xml.transform package 23
- javax.xml.validation package 24
- javax.xml.xpath package 25
- javax.xml package 27
- JAXB 31
- JAXB commands 33
- JAXB-defined package 32
 - API 32
- JAXB functionality 236
- JAXP 20
 - XSLT features 105
- JAXP-defined package 21
 - API 21
 - javax.xml 27
 - javax.xml.datatype 26
 - javax.xml.namespace 26
 - javax.xml.parsers 21
 - javax.xml.sax 30
 - javax.xml.stream 22
 - javax.xml.stream.events 23
 - javax.xml.stream.util 23
 - javax.xml.transform 23
 - javax.xml.transform.dom 24
 - javax.xml.transform.sax 24
 - javax.xml.transform.stax 24
 - javax.xml.transform.stream 24

- javax.xml.validation 24
- javax.xml.xpath 25
- org.w3c.dom 27
- org.w3c.dom.bootstrap 28
- org.w3c.dom.events 29
- org.w3c.dom.ls 28
- org.xml.sax.ext 30
- org.xml.sax.helpers 30

K

- know-how for efficiently using XML Processor 237

L

- list of command 33

M

- merits and demerits of each parser 235
- MS932 17

N

- namespace URI (Apache-specific) 85
- note
 - Cosminexus XML Processor 148
 - DOM parser 153
 - high-speed parse support function 196
 - implementation-dependent specification 192
 - javax.xml.datatype package 180
 - javax.xml.validation package 181
 - javax.xml.xpath package 183
 - JAXB 199
 - org.w3c.dom.ls package 188
 - org.w3c.dom.bootstrap package 187
 - org.w3c.dom package 185
 - org.xml.sax.ext package 190
 - parse performance 198
 - runtime 216
 - SAX parser 154
 - schema compiler 199
 - schema generator 207
 - schema validation 168
 - XInclude 191
 - XSLT 172
 - XSLT and XSLTC 169
 - XSLTC 173
- note on high-speed parse support function
 - how to specify an absolute path 197

- Pre-Parse XML document 197
- setting up preparsed object 196
- tuning information 197
- XInclude 196
- XML1.1 196
- XML parser using preparsed object 196

- note on XSLT

- case where XSLT does not report errors 172

- note on XSLTC

- case where XSLTC does not report errors 177
 - others 179
 - stylesheet size 173
 - transformation performance 173
 - XPath expression 177
 - XSLT element 174

- note on XSLTC 173

- notes

- schema cache functionality 194
- notes (General XML Processor) 222
- notes common to JAXP1.4 function 149

O

- org.w3c.dom.bootstrap package 28
- org.w3c.dom.events package 29
- org.w3c.dom.ls package 28
- org.w3c.dom package 27
- org.xml.sax.ext package 30
- org.xml.sax.helpers package 30
- org.xml.sax package 30
- overview
 - Cosminexus XML Processor 10
 - XSLTC transformer 42
- overview of schema cache functionality 45

P

- package name
 - Cosminexus XML Processor 119
- parser switching functionality 87
- parsing
 - XML document 69
- positioning
 - Cosminexus XML Processor 14
- precautions related to schema cache functionality 194
 - precautions related to error output 194
 - precautions related to performance 194

- precautions related to reset up and deletion of cache 195
- precautions related to schema definition files 194
- prepared object 57
- PreparsedObject class 73
- PreparsedObjectFactory class 70
- Pre-Parse XML document 58
- Procedure for creating a program using Cosminexus XML Processor 118
- product feature
 - Cosminexus XML Processor 11
- program creation procedure 118
- program execution
 - troubleshooting 120
- property (Apache-specific) 85
- property value 109

S

- sample program
 - DOM parser 121, 137
 - flow of processing using XML schema 131
 - procedure for using DOM parser 121
 - procedure for using XSLT transformer 143
 - SAX parser 126, 139
 - XML schema 131
 - XSLTC transformer 147
 - XSLT transformer 142
- SAX2 features
 - Cosminexus XML Processor 102
 - how to use 102
- SAX2 features and properties
 - how to use 102
- SAX2 properties
 - Cosminexus XML Processor 102
 - how to use 103
- SAX parser 235
 - general procedure for creating sample program 126
 - how to use Shift_JIS switch function 110
 - note 154
 - sample program 126, 139
- schema cache functionality 45
- schema document
 - personalData.xsd 136
 - purchaseOrder.xsd 134
- schema validation
 - note 168
- setting up

- prepared object 69
- Shift_JIS switch function 17, 109
- SJIS 17
- specification
 - JAXB 16
- StAX parser 235
- support range
 - function assumed as vendor-specific according to JAXB specification 232
 - JAXB function 229
 - JAXB specification 229
- support range for JAXB characters 230
- system development and operation
 - XSLTC transformer 42

T

- TransformerFactoryXSLTC class 43
- translet 42
- troubleshooting
 - program execution 120
- tuning details file 76
- tuning summary file 76
- type of schema document to identify
 - attribute to use 107

V

- validation functionality 236

W

- W3C specification
 - error message in XML schema 86
- whitespace 176

X

- XML document
 - how to set schema document 107
 - purchaseOrder.xml 132
 - purchaseOrder-fail.xml 133
- XML parser and XSLT transformer
 - available function 15
- XML schema
 - flow of processing for creating sample program 131
 - sample program 131
- XML schema property
 - how to use 106
- XPath functionality 236

XSLT

- differences in behaviors [224](#)

- note [172](#)

XSLT and XSLTC

- note [169](#)

XSLTC

- note [173](#)

XSLTC transformer

- class [43](#)

- how to use [44](#)

- overview [42](#)

- sample program [147](#)

- system development and operation [42](#)

XSLTC transformer function [42](#)

- changes to program [147](#)

XSLT features

- JAXP [105](#)

XSLT functionality [236](#)

XSLT transformer

- general procedure for creating sample program [143](#)

- how to use Shift_JIS switch function [111](#)

- sample program [142](#)