

# Intel® XSLT Accelerator 1.1 for Java\* Environments

User's Guide

---

*April 2007*

Document Number: 315825-002US

World Wide Web: <http://developer.intel.com>

Version	Version Information	Date
315825-001US	Initial version	12/2006
315825-002US	Updates for 1.1 release	04/2007

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

The software described in this document may contain software defects which may cause the product to deviate from published specifications. Current characterized software defects are available on request.

This guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Developers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Improper use of reserved or undefined features or instructions may cause unpredictable behavior or failure in developer's software code when running on an Intel processor. Intel reserves these features or instructions for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from their unauthorized use.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

\* Other names and brands may be claimed as the property of others.

Copyright © 2006-2007 Intel Corporation. All rights reserved.

# Contents

---

## Chapter 1

### Overview

About This Document .....	1-1
Purpose .....	1-1
Intended Audience .....	1-1
Using this Document .....	1-1
Conventions and Symbols .....	1-2
Acronyms and Definitions .....	1-3
Related Information .....	1-3
Product Overview .....	1-4
About XSL Transformation .....	1-4
Supported Environments .....	1-5
Major Features .....	1-6
Standard Java* API .....	1-7

## Chapter 2

### Quick Start

Before you Begin .....	2-1
Checking your Installation .....	2-1
Configuring the Environment .....	2-4
Switching on Intel XSLT Accelerator .....	2-5

Getting Started with Intel XSLT Accelerator .....	2-5
Examples .....	2-6
About Examples .....	2-6
Running Examples .....	2-7

### **Chapter 3**

#### **Using Intel® XSLT Accelerator**

Integrating Intel XSLT Accelerator.....	3-1
Transforming Data .....	3-1
Performing the XSL Transformation .....	3-1
Configuring Output .....	3-3
Using Object Types .....	3-4
XSLT Extensions .....	3-4
EXSLT Extensions .....	3-5
User-Defined Extension Functions .....	3-8
Other Extensions .....	3-12

### **Chapter 4**

#### **Internal Characteristics**

Threading Support.....	4-1
Performance .....	4-1

### **Chapter 5**

#### **Troubleshooting**

Installation Checks .....	5-1
Running the examples.....	5-2

# Overview

---

# 1

This part defines the audience and purpose of the manual, provides instructions on its usage, and gives a product introduction with its key features and specifics.

## About This Document

### Purpose

This document describes Intel® XSLT Accelerator for Java\* Environments with an overview of major features and a high-level view of API and intended usage. With this document, you will be able to start using Intel XSLT Accelerator, run the samples supplied with the package, employ advanced functionality of the library, and troubleshoot any product-related issues.

### Intended Audience

The document is targeted at developers of JAXP-based Java\* environments (JRE\* or JDK\*) that desire higher XSLT processing performance, as well as Java developers that need to use XSLT transformation in their applications.

The document assumes knowledge of XML and XSLT basics, and Java programming skills.

### Using this Document

The guide consists of the following chapters:

- [“Quick Start”](#) - post-installation checks and information on how to get started with using the library

- [“Using Intel® XSLT Accelerator”](#) - detailed instructions on intended usage scenarios
- [“Internal Characteristics”](#) - specifics of the current implementation, such as performance data, threading and memory characteristics
- [“Troubleshooting”](#) - tips on typical mistakes to avoid when using Intel XSLT Accelerator

This guide is distributed in PDF format and in web-based HTML format for easier browsing and navigation. To see the web-based version, go to `index.html` in the `doc/webhelp` folder.

Please note that when viewing this guide in PDF format, your Adobe\* Reader\* viewer might generate hyperlinks from URL-formatted text of the guide. To disable this option, edit your preferences to disable automatic hyperlink generation.

## Conventions and Symbols

This section lists the conventions and symbols used in this document.

Conventions, symbols, and metacharacters used in this online help are described in Table 1-1.

**Table 1-1 Conventions and Symbols used in this Online Help**

Convention	Explanation	Example
<i>Italic</i>	Italic is used for the introduction of new terms, denotation of terms, placeholders, titles of manuals.	<i>XSL transformation</i> is a form of XML data processing. <i>String</i> functions are responsible for.. The bitmap is <i>width</i> pixels wide and <i>len</i> pixels high For more information, refer to the <i>Intel® Linker Manual</i> .
Monospace	Indicates filenames, directory names and pathnames, commands and command-line options, function names, classes in body text.	<code>xalan.jar</code> <code>lib\intel-xslt.jar</code> <code>echo \$PATH</code> Reads the Template objects...
<b>Monospace bold</b>	Indicates what you type as input on a command line.	<code>PATH="install_dir/lib"</code>

## Acronyms and Definitions

This section lists the acronyms used throughout the document with their definitions.

**Table 1-2 Acronyms Used in this Document**

Acronym	Definition
API	Application programming interface
DOM	Document object model
EXSLT	Extensions to XSL Transformation
I/O	Input and output
JAXP	Java* API for XML processing
JDK*	Java* development kit
JNI	Java* Native Interface
JRE*	Java* Run-time Environment
SAX	Simple API for XML
TrAX	Transformation API for XML processing
URI	Unique resource identifier
XML	Extensible markup language
XSL	Extensible stylesheet language

## Related Information

This section lists reference to relevant external documents referenced in the current document.

1. XML 1.0 Recommendation, <http://www.w3.org/TR/xml/>
2. XSL 1.0 Recommendation, <http://www.w3.org/TR/xsl/>
3. Java\* API for XML Processing (JAXP) 1.3 description, <http://java.sun.com/webservices/jaxp/index.jsp>
4. Apache Xalan\* Java\* transformer, <http://xml.apache.org/xalan-j/>
5. Apache\* XSLTC transformer, <http://xml.apache.org/xalan-j/xsltc/index.html>
6. Extensions to XSLT, <http://exslt.org/>
7. OASIS\* XSLT Conformance test suite, [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=xslt](http://www.oasis-open.org/committees/documents.php?wg_abbrev=xslt)
8. Java\* Signal Chaining, <http://java.sun.com/j2se/1.5.0/docs/guide/vm/signal-chaining.html>

## Product Overview

XML is widely used as a common language that facilitates data exchange [1]. Representing data in a flexible and dynamic way, XML enables smooth and seamless communication from one application to another and does not depend on hardware or software infrastructure.

Employing XML relies on *XML processors* that read an XML document, process it, and hand the results over to the application. This way, XML processors provide the application program with XML document contents in the form that is easy to apply for further work with the XML document. Transformation can involve representation of the complete document into XML or another format, re-arranging and editing XML data in the output or changing the presentation of the data.

However, the flexibility of XML comes with a price of a complex processing and increased CPU work load. As a result, several processing bottlenecks have been introduced to applications employing XML processing.

Intel® XSLT Accelerator is a software product for accelerating XML transformations in Java\* based application server environments. This Java-based solution provides extensible stylesheet language transformation (XSLT) functionality for XML data. The Intel XSLT Accelerator library complements existing XML infrastructures by providing additional performance on Intel architecture based platforms.

## About XSL Transformation

*XSL transformation* is a form of XML data processing that instructs the XML processor on data transformation in the extensible stylesheet language (XSL) format [2]. With this transformation, you can add/remove elements and attributes to or from the output file, define its format and appearance. For example, with XSL transformation, you can convert a set of XML data from your application into an HTML page displayed to the user.

Intel® XSLT Accelerator for Java\* Environments is a speedy transformer that supports multiple XML processing models. This library employs XSL transformation to convert textual data and performs other conversions, such as operations with events and sets of nodes.

In addition to XSL transformation, Intel XSLT Accelerator can perform *identity transformation*, where the data format is changed, but the data is not altered. A subset of identity transformation is *serialization*, the operation of transforming a hierarchical set of XML nodes into a stream.

## Supported Environments

Intel® XSLT Accelerator for Java\* Environments operates on 5.0 Java\* development kits, Sun Java\* 2 Standard Edition and BEA JRockit\* R26. The Java sources are built using the javac tool 1.5.0.

Intel XSLT Accelerator runs on Windows\* and Linux\* operating systems on IA-32 and Intel® 64 architectures. The following tables list the platforms supported in this release:

**Table 1-3 Supported Platforms<sup>1</sup>**

Version	Architecture	
	IA-32	Intel® 64
<b>Linux* systems</b>		
Red Hat Enterprise Linux* AS 4.0 - 2.6 kernel	x	x
Red Hat Enterprise Linux* ES 4.0 - 2.6 kernel	x	x
Red Hat Enterprise Linux* AS 3.0 - 2.4 kernel	x	x
Red Hat Enterprise Linux* ES 3.0 - 2.4 kernel	x	x
SUSE* Linux Enterprise Server 10 - 2.6 kernel	x	x
SUSE* Linux Enterprise Server 9 - 2.6 kernel	x	
<b>Windows* systems</b>		
Microsoft Windows Server 2003*	x	x
Microsoft Windows Vista*	x	x
Microsoft Windows Vista* Server "Longhorn"	x	x

1. Intel® XSLT Accelerator runs on AMD\* processors that support listed operating systems.



**NOTE.** In the current version, Intel XSLT Accelerator is 32-bit only and runs on the Intel® 64 architecture in the 32-bit compatibility mode. Intel XSLT Accelerator is not supported on the Intel® Itanium® architecture. Best performance is obtained on Dual-Core Intel® Xeon® processor 7100 Series.

## Major Features

Intel® XSLT Accelerator for Java\* Environments has the following major features:

- *High Performance:* Intel XSLT Accelerator demonstrates high transformation performance by utilizing the underlying native XML processing core; see section [“Performance”](#). In the product, major processing is done inside the core part that interacts with the client application via the standard Java\* interface; see section [“Inside Intel XSLT Accelerator”](#).
- *Conformance and Compatibility:* The process of transformation is defined by a number of standards. Intel XSLT Accelerator fully complies with the XML 1.0 [1] and XSL 1.0 [2] standards.

To expand the number of supported features, Intel XSLT Accelerator has been made fully compatible with the Apache\* Xalan\* XSLTC [5] processor. The current version also supports certain features of the Apache\* Xalan-Java\* [4] processor. Specifically, Intel XSLT Accelerator supports Xalan-Java specific XML namespace declaration formats; see section [“Namespace Declaration for Extension Functions”](#).

The current version of the Intel XSLT Accelerator library successfully passes the OASIS\* conformance test suite [7] for the Apache Xalan-Java processor.

- *Extension Functions Support:* The library supports XSLT extension function mechanisms to allow applications to extend XSLT functionality through Java extension functions [6]. In addition, the library supports several sets of EXSLT extension functions. For details on supported functions and guidelines on their usage, see section [“EXSLT Extensions”](#).
- *Varied Input and Output:* The library supports various XSL textual transformations of XML-formatted data and conversion of XML to HTML and vice versa. The library supports a large number of character sets with optimized encoding and decoding processing.

Additionally, the library supports transformation of XML input data in the format of data streams, Document Object Model (DOM) trees or Simple API for XML (SAX) events, and can produce output in the same formats. For specifics of each I/O type, see section [“Using Object Types”](#).



---

**NOTE.** Processing files of over 1GB in size might cause unexpected transformation failures.

---

## Standard Java\* API

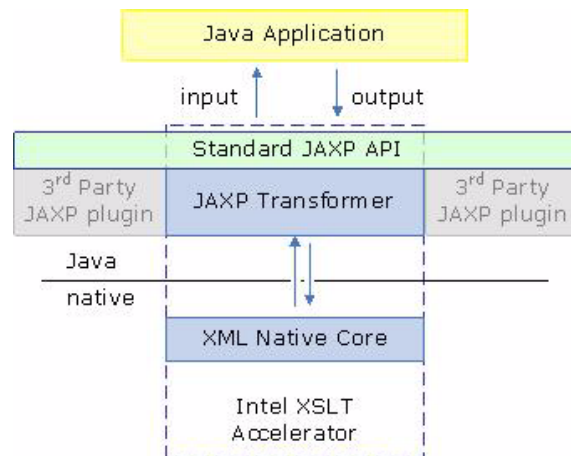
The Java\* API for XML Processing, JAXP [3], is a de-facto standard for XML processing in the Java environments and is currently included into the Java 1.5.0 distribution. This API consists of a number of layers, such as XML standards, XSLT transformation, Simple API for XML (SAX), and so on.

### JAXP Interface

Intel® XSLT Accelerator for Java\* Environments implements a part of JAXP version 1.3, specifically, the transformation API for XML processing (TrAX). As a result, you can migrate to the new Intel XSLT Accelerator with minimal effort. By conforming to the JAXP standard interface, Intel XSLT Accelerator shields application developers from implementation details and provides a significant performance boost at same time.

The figure below shows the relationship between the JAXP environment and Intel XSLT Accelerator.

**Figure 1-1 Intel XSLT Accelerator in the JAXP-based Environment**



At the code level, the JAXP architecture allows easy replacement of the underlying implementation through the use of abstract Factory classes with static `newInstance()` methods. Specifically, JAXP provides an abstract `javax.xml.transform.TransformerFactory` class that allows a concrete Factory object to be created from the static function `TransformerFactory.newInstance()`. This concrete Factory object is a wrapper around the Intel implementation of the XML transformer engine. Details on how to configure JAXP environment for use of Intel XSLT Accelerator is described in section [“Configuring the Environment”](#).

## Inside Intel XSLT Accelerator

As shown in [Figure 1-1](#), Intel® XSLT Accelerator consists of two major parts: the *Java XML Transformer* that exports the JAXP interface, and the underlying *native XML core* responsible for XML processing. These components interact by using the Java Native Interface (JNI).

By moving transformation to the native core, Intel XSLT Accelerator speeds processing with the help of:

- Intel compiler optimizations specific to the Intel® Core™2 Duo processor family
- The multithread fine-grain locking mechanism
- Efficient cache line management

These steps are not available from Java alone. The combination of these steps result in highest XSLT performance amongst XSLT Processors with Java interface. Consult the section [“Performance”](#) for specific data about Intel XSLT Accelerator performance characteristics.

# Quick Start

---

# 2

This part enables you to start using the product quickly by providing installation and usage instructions. For in-depth usage information, consult [“Using Intel® XSLT Accelerator”](#).

## Before you Begin

Before you begin using Intel® XSLT Accelerator for Java\* Environments, you must install this library, as shown in the `Install.html` document.

After installing Intel XSLT Accelerator, check that your installation has completed successfully and configure your environment, as described below.

## Checking your Installation

Check your installation directory against the description in this section. If an element is missing, your install might be incomplete. Consult the section [“Troubleshooting”](#) for recommendations on resolving the issue.

### On Windows\*

After a successful install, the following directory tree appears under the relative installation directory `IntelXsltJ`:

<code>IntelXSLTAcceleratorEULA.pdf</code>	Your license agreement for using Intel® XSLT Accelerator for Java* Environments
<code>*-LICENSE.txt</code>	All license files applicable to the bundled product
<code>redist.txt</code>	The list of files that are redistributed with a distribution license of the product
<code>apache_2_0_NOTICE.txt</code>	The Apache Software Foundation* notice file that accompanies redistributed Apache* code and documentation

bin	Shared Intel XSLT Accelerator native library <ul style="list-style-type: none"><li>intel-xslt.dll</li></ul>
doc	Product documentation <ul style="list-style-type: none"><li>README</li><li>INSTALL.html</li><li>ReleaseNotes.html</li><li>doc_index.htm</li><li>Intel XSLT Accelerator for Java User Guide.pdf</li><li>webhelp Directory with the HTML version of the user guide.</li><li>javadoc Directory with HTML documentation of the Java* interface; specifically, the packages <code>javax.xml.transform.*</code>; <code>org.w3c.dom*</code>; <code>org.xml.sax*</code></li></ul>
examples	Sample Java* code <ul style="list-style-type: none"><li>TransformExample</li><li>RedirectExample</li><li>ExtensionFunctionExample</li><li>ApacheXalanSamples<ul style="list-style-type: none"><li>ApplyXPathJAXP</li><li>DOM2DOM</li><li>Pipe</li><li>SAX2SAX</li><li>serializer.jar</li><li>SimpleTransform</li><li>TransformThread</li><li>trax</li><li>UseStylesheetParam</li><li>UseStylesheetPI</li><li>UseXMLFilters</li><li>Validate</li></ul></li></ul>
lib	Java jar files for this product; see license files in the root directory <ul style="list-style-type: none"><li>intel-xslt-1_1_0.jar – Intel XSLT Accelerator jar file.</li></ul> In the evaluation version of the library, you get the file <code>intel-xslteval-1_1_0.jar</code> .

## On Linux\*

After a successful install, the following directory tree appears under the relative installation directory `IntelXsltJ`:

<code>IntelXSLTAcceleratorEULA.pdf</code>	Your license agreement for using Intel® XSLT Accelerator for Java* Environments
<code>*-LICENSE.txt</code>	All license files applicable to the bundled product
<code>redist.txt</code>	The list of files that are redistributed with a distribution license of the product
<code>apache_2_0_NOTICE.txt</code>	The Apache Software Foundation* notice file that accompanies redistributed Apache* code and documentation
<code>bin</code>	Shared Intel XSLT Accelerator native library <ul style="list-style-type: none"><li><code>libintel-xslt.so</code></li></ul>
<code>doc</code>	Product documentation <ul style="list-style-type: none"><li><code>README</code></li><li><code>INSTALL.html</code></li><li><code>ReleaseNotes.html</code></li><li><code>doc_index.html</code></li><li><code>Intel XSLT Accelerator for Java User Guide.pdf</code></li><li><code>webhelp</code> Directory with the HTML version of the User Guide</li><li><code>javadoc</code> Directory with HTML documentation of the Java* interface; specifically, the packages <code>javax.xml.transform.*</code>; <code>org.w3c.dom*</code>; <code>org.xml.sax*</code></li></ul>
<code>examples</code>	Sample Java code <ul style="list-style-type: none"><li><code>TransformExample</code></li><li><code>RedirectExample</code></li><li><code>ExtensionFunctionExample</code></li><li><code>ApacheXalanSamples</code><ul style="list-style-type: none"><li><code>ApplyXPathJAXP</code></li><li><code>DOM2DOM</code></li><li><code>Pipe</code></li><li><code>SAX2SAX</code></li><li><code>serializer.jar</code></li><li><code>SimpleTransform</code></li><li><code>TransformThread</code></li><li><code>trax</code></li><li><code>UseStylesheetParam</code></li></ul></li></ul>

- UseStylesheetPI
- UseXMLFilters
- Validate

lib                    Java jar files for this product; see license files in the root directory

- intel-xslt-1\_1\_0.jar – Intel XSLT Accelerator jar file

In the evaluation version of the library, you get the file intel-xslteval-1\_1\_0.jar

## Configuring the Environment

To start working with Intel® XSLT Accelerator for Java\* Environments, you may need to set up environment variables, as described in this section. You might want to save a backup copy of the original environment files before altering the path settings.

### On Windows\*

After installation, your system should be ready to start work with Intel XSLT Accelerator. To ensure correct operation, check that the environment variables are set properly:

```
echo $PATH
echo $CLASSPATH
```

The PATH variable should include the installation directory of Intel XSLT Accelerator, and the CLASSPATH variable should point to the file intel-xslt-1\_1\_0.jar in the lib\ subdirectory of the installation location.

If your variable settings are incorrect, configure them to point to the following:

```
PATH="installation_dir\bin;%PATH%"
CLASSPATH="installation_dir\lib\intel-xslt-1_1_0.jar;%CLASSPATH%"
```

### On Linux\*

To start working with Intel XSLT Accelerator, enable Java Signal Chaining [8] and configure the environment variables to point to the following locations:

```
LD_LIBRARY_PATH="installation_dir/bin:$LD_LIBRARY_PATH"
CLASSPATH="installation_dir/lib/intel-xslt-1_1_0.jar:$CLASSPATH"
```

You can also specify the location of the transformer library at run time when starting the Java\* application, as follows:

```
-Djava.library.path="installation_dir/bin"
```

---

## Switching on Intel XSLT Accelerator

To enable Intel® XSLT Accelerator for Java\* Environments, edit the system property setting to point to the correct implementation in one of the following ways:

- Edit `jaxp.properties` in the `lib` directory inside your JDK\* directory, as follows:  
`javax.xml.transform.TransformerFactory=com.intel.xml.transform.TransformerFactoryImpl`
- Specify the command-line argument to Java\*, as follows:  
`-Djavax.xml.transform.TransformerFactory=com.intel.xml.transform.TransformerFactoryImpl`  
For example, on Windows\* type:  

```
java  
-Djavax.xml.transform.TransformerFactory=com.intel.xml.transform.TransformerFactoryImpl -Djava.library.path=../../bin -cp  
./../../lib/intel-xslt-1_1_0.jar SimpleTransform
```
- In your Java application, invoke the following method:  

```
System.setProperty("javax.xml.transform.TransformerFactory",  
"com.intel.xml.transform.TransformerFactoryImpl");
```

## Getting Started with Intel XSLT Accelerator

This section gives you a quick start with transforming your data. Follow the instructions to switch your application to Intel® XSLT Accelerator for Java\* Environments. Changes are minimal due to the standard JAXP interface and the pluggability layer. For a detailed description of the process, see [“Transforming Data”](#).

To get started with Intel XSLT Accelerator, do the following:

1. Ensure that you have installed the library on your system as described in the `INSTALL` guide.
2. Check configuration settings as shown in [“Configuring the Environment”](#) and [“Switching on Intel XSLT Accelerator”](#).
3. From within your application, do the following:
  - a. Use the transformer factory to instantiate a `Transformer` that will work with the stylesheet you specify.
  - b. Launch the transformer process.

## Examples

The current distribution of Intel® XSLT Accelerator includes several source code examples that show how you can use the product with your application. For a definition of supplied examples and usage instructions, consult this section.

## About Examples

The examples provided with Intel® XSLT Accelerator cover various types of transformation functionality that is available with the product. The examples are in the `IntelXsltJ/examples` directory, with corresponding `xml` and `xsl` source files, and any other files required for the illustrated type of transformation. Each group of examples is located in a separate directory.

### TransformExample

This directory includes an example of running stream-to-stream transformation for the specified `xml` and `xsl` files.

### ExtensionFunctionExample

This directory contains an example of employing extension functions defined in the Java\* language during XSLT transformation. Specifically, the stylesheet in this example invokes Java methods defined in `java.lang.Integer` and `java.lang.String` classes. For more information on using extensions with Intel XSLT Accelerator, see [“XSLT Extensions”](#).

### RedirectExample

This directory contains an example of using the Xalan-specific `redirect` extension. The example shows that instead of writing the entire result of transformation into a single `javax.xml.transform.Result` object, you can divide the output into portions and write them to the specified files. For more details on the example, consult the Apache\* Xalan-Java\* website [\[4\]](#).

### ApacheXalanSamples

This directory holds selected samples from the Apache\* Xalan-Java\* web site [\[4\]](#) that show ease of adapting your Apache Xalan-Java based application to Intel XSLT Accelerator. Each example is located in a separate subdirectory. With this release, the following examples are supplied:

<code>SimpleTransform</code>	uses the <code>xsl</code> stylesheet to convert an <code>xml</code> source file and print the output.
<code>Trax</code>	demonstrates the use of the Transformation API for XML processing (TrAX).

<code>SAX2SAX</code>	explicitly sets the <code>SAX XMLReader</code> and <code>SAX ContentHandler</code> for processing the <code>xsl</code> stylesheet and the <code>xml</code> input files, and producing the output.
<code>DOM2DOM</code>	processes an input <code>DOM Document</code> and creates an output <code>DOM</code> that is ready for further processing.
Chain transformations	feed output of one transformation as input into another: <ul style="list-style-type: none"><li>• <code>Pipe</code> illustrates a forwarding model where each <code>Transformer</code> object directs its output into the following transformer.</li><li>• <code>UseXMLFilters</code> illustrates a backward model where each <code>Transformer</code> object is an extension of the <code>SAX XMLFilter</code> interface and sets each preceding filter as the parent of the following filter in the chain.</li></ul>
<code>TransformThread</code>	spawns multiple threads, with each thread running two transformations on two different XML files.
<code>UseStylesheetPI</code>	reads the stylesheet processing instruction in the XML source file to select the stylesheet for the transformation. For more information, see <a href="#">“Configuring Output”</a> .
<code>UseStylesheetParam</code>	takes a stylesheet parameter, reads the <code>xml</code> source file and the <code>xsl</code> stylesheet, and performs the transformation. The stylesheet parameter appears as a text node in the output. For more information on these parameters, see <a href="#">“Adding Stylesheet Parameters”</a>
<code>ApplyXPathJAXP</code>	uses the XPath API in JAXP 1.3 [3] to evaluate an XPath expression against an XML document and return the evaluation result in the specified type.
<code>Validate</code>	consists of two parts: <ul style="list-style-type: none"><li>• <code>ValidateXMLInput</code> turns on validation, parses the XML input and reports errors and warnings to a SAX event handler.</li><li>• <code>Validate utility</code> uses a SAX event handler to verify that input XML files conform to their declared document type, are well-formed and valid.</li></ul>

In addition to example subdirectories, the `ApacheXalanSamples` directory contains the supporting file `serializer.jar`.

## Running Examples

To run an example, do the following:

1. Check that you have set up the environment as described in the section [“Configuring the Environment”](#).
2. On the command line, navigate to the corresponding directory.
3. Compile the source files of the example by running:  

```
javac -cp ../../serializer.jar *.java
```

Where \* stands for the example filename.
4. Run the example using the format of:  

```
java classname args
```

Where classname is the example class file, and args includes run-time arguments, if any.  
The commands below show how to compile and execute the stream-to-stream transformation and the SAX2SAX examples.

### Example 2-1 Running the SAX2SAX Example

---

```
cd ApacheXalanSamples/SAX2SAX
javac -cp ../../serializer.jar *.java
java
-Djavax.xml.transform.TransformerFactory=com.intel.xml.transform.Transform
erFactoryImpl -Djava.library.path=../../bin -cp
../../lib/intel-xslt-1_1_0.jar;../../serializer.jar SAX2SAX
```

---

### Example 2-2 Running the Transform Example

---

```
cd TransformExample
javac TransformExample.java
java
-Djavax.xml.transform.TransformerFactory=com.intel.xml.transform.Transform
erFactoryImpl -Djava.library.path=../../bin -cp
../../lib/intel-xslt-1_1_0.jar; TransformExample foo.xml foo.xml
out.txt
```

---

You can modify the source of the provided examples or you can compile the example's class right away and place it in the CLASSPATH variable.

# Using Intel® XSLT Accelerator

---

## 3

This part describes usage and work model of Intel® XSLT Accelerator for Java\* Environments in detail, including implementation specifics, configuration facilities, and extensions support.

## Integrating Intel XSLT Accelerator

As described in the section [“Standard Java\\* API”](#), Intel® XSLT Accelerator is a software library that can be seamlessly integrated into a JAXP-based Java\* environment. This section defines the steps that you need to take to make your Java application code interact with Intel XSLT Accelerator via this standard API.

To start working with the processing library, do the following:

1. Install the library and verify installation completion as described in section [“Checking your Installation”](#).
2. Configure your environment as described in section [“Configuring the Environment”](#).

## Transforming Data

This section describes transformation of XML data with Intel XSLT Accelerator and facilities for customizing the process.

## Performing the XSL Transformation

To transform your input data with Intel® XSLT Accelerator, do the following:

1. Instantiate a `TransformerFactory` object.  
The JAXP interface enables you to abstract your application code from the internals of the transformer. For that, an abstract `TransformerFactory` class is used with a static method `newInstance()`. This method does the following:

1. Reads the settings, see section [“Configuring the Environment”](#).
  2. Selects the underlying transformer implementation.
  3. Wraps the selected tranformer with a concrete `Factory` object.
2. Feed your input data into the transformer.
- The method `newTransformer(Source xslSource)` in the `TransformerFactory` class provides the new `Tranformer` object. This method reads the supplied stylesheet `Source`, produces a `Templates` object out of it and returns the `Transformer` object that you can use.
- You can supply the stylesheet just as your input data: in a stream of XML markup (`StreamSource`), in a DOM node (`DOMSource`) or SAX input (`SAXSource`).
3. Perform the transformation.
- Method `transform(Source xmlSource, Result transformResult)` of the `Transformer` object works over the XML source and places output in a `Result` object. Specifically, for each node in the XML source, the method does the following:
1. Reads instructions in the `Templates` objects
  2. Selects the template to apply out of the following choices:
    - A template rule in the `Templates` object
    - The default template rule as specified in the XSLT spec
    - None
- The XML source can go in the form of a `StreamSource`, `DOMSource`, or `SAXSource` object, and the output can be a `StreamResult`, `DOMResult`, or `SAXResult` object as specified in the template rule.

XSL transformation is illustrated by the `TransformExample` sample application that is supplied with the product. Consult [“Examples”](#) for details on finding and using the example.

---

**Example 3-1 Using a transformer to get a DOM tree**

---

```
// Generate a Transformer object.
tFactory = TransformerFactory.newInstance();
javax.xml.transform.Transformer transformer =
tFactory.newTransformer
    (new javax.xml.transform.stream.StreamSource("foo.xml"));
// Create an empty DOMResult object for the output.
javax.xml.transform.dom.DOMResult domResult =
    new javax.xml.transform.dom.DOMResult();
// Perform the transformation.
transformer.transform(new
javax.xml.transform.dom.DOMSource(inDoc),
    domResult);
// Now you can get the output Node from the DOMResult.
org.w3c.dom.Node node = domResult.getNode();
```

---

You can also use an extension function to redirect your output into one or more files, see [“Redirect Extension”](#).

## Configuring Output

Output properties are an API feature that exposes properties specified in the `xsl:output` element of a stylesheet. After a stylesheet is compiled, the output properties that are specified in the XSLT stylesheet can be queried using the function `getOutputProperties()` on the `java.xml.transform.Templates` or the `javax.xml.transform.Transformer` object. You can override output property values when performing a transformation by calling functions `setOutputProperty()` and `setOutputProperties()` on the `Transformer` object.

### Adding Stylesheet Parameters

Instructions in your stylesheet can get certain parameters during the transformation. For that, the `setParameter()` method of the `Transformer` object is used. After setting a parameter, you can retrieve it using the `getParameter()` method.

## Using Object Types

As shown in the transformation description, XML processing can involve data streams, DOM trees and SAX events. Each object type has its specifics in the transformation, as described below.

### Streams

To specify a `StreamSource` object, use a system ID, a filename following the URI syntax, and a `java.io.InputStream` or `java.io.Reader` object.

### DOM Trees

Your transformations can involve `DOMSource` and `DOMResult` objects provided by the `javax.xml.transform.DOM` package. These involve operations with a DOM (document object model) tree. To transform `DOMSource` into a stream, create a new `Transformer` object and make a “copy” of your DOM tree as a stream of data.

To produce a `DOMResult` object out of a stream of data, use the `DocumentBuilderFactory` object, which creates a `DocumentBuilder` object for your needs. To transform your data into a DOM tree, create a new `DOMResult` object or use `DOMResult.setNode()` to assign a new container.

### SAX Events

You can use SAX (Simple API for XML) events in your input data, source stylesheet instructions or output. In the transformation engine, the `SAXParser` interface defines several `parse()` methods to handle SAX events. When a `parse()` method is called, the parser invokes one of the callback handler methods in your application.

You can implement content handlers, error handlers, and other methods depending on your needs. Because `SAXParser` is a wrapper for the `SAXReader` object, you can easily plug in your own reader instead of it.

For SAX-specific methods, you can use a `SAXTransformerFactory` object [3]. This feature enables the use of XML filter to pass output of one transformation as input for another transformation via the `SAXTransformerFactory.newXMLFilter(Source)` and `newXMLFilter(Templates)` methods.

## XSLT Extensions

You can expand the functionality of XSLT transformations through the use of XSLT extensions. An extension can be a function or an element. The Intel® XSLT Accelerator library provides a subset of extension functions, as defined by the EXSLT community project [6]. Further in the document, these functions are referred to as *EXSLT extensions*.

Extensions can also be user-defined, that is, provided by the end-user code. These are limited to extension functions. Intel XSLT Accelerator supports user-defined extension functions implemented in the Java\* programming language. This way, an extension function is a Java method that is invoked during the execution of an XSLT transformation. The method can be static or instance based.

This section provides an overview of supported extensions and their usage.

## EXSLT Extensions

The current version of Intel® XSLT Accelerator for Java\* Environments supports EXSLT extensions grouped into the following modules:

- *Common* functions cover the basic operations; for example, the `exsl:object-type` function returns the type of the supplied object.

The current version of Intel XSLT Accelerator supports all functions of this module defined in the EXSLT resource [6], specifically, the `exsl:node-set` and `exsl:object-type` functions. Additionally, the library supports the `nodeset` Xalan-Java extension, which matches the `common:node-set` EXSLT extension function.

- *Dates and times* functions handle operations related to date and time; for example, the `date:hour-in-day` function returns the hour of the day as a number.

The current version of Intel XSLT Accelerator supports all core and most other functions of this module defined in the EXSLT resource [6], specifically:

- `date:date-time`
- `date:date`
- `date:time`
- `date:year`
- `date:leap-year`
- `date:month-in-year`
- `date:month-name`
- `date:month-abbreviation`
- `date:week-in-year`
- `date:day-in-year`
- `date:day-in-month`
- `date:day-of-week-in-month`
- `date:day-in-week`
- `date:day-name`

- `date:day-abbreviation`
- `date:hour-in-day`
- `date:minute-in-hour`
- `date:seconds`
- `date:second-in-minute`
- `date:format-date`
- `date:difference`
- `date:add`
- `date:add-duration`
- `date:duration`

The current version of the library does not implement the following EXSLT functions of this module: `date:parse-date`, `date:week-in-month`, and `date:sum`.

- *Math* functions provide facilities for performing mathematical operations; for example, the `math:max` function returns the maximum value of the nodes passed as the argument.

The current version of Intel XSLT Accelerator supports all functions of this module defined in the EXSLT resource [\[6\]](#), specifically:

- `math:min`
- `math:max`
- `math:highest`
- `math:lowest`
- `math:abs`
- `math:sqrt`
- `math:power`
- `math:constant`
- `math:log`
- `math:random`
- `math:sin`
- `math:cos`
- `math:tan`
- `math:asin`
- `math:acos`

- `math:atan`
- `math:atan2`
- `math:exp`
- *Sets* functions allow you to manipulate node sets; for example, the `set:difference` function gets nodes of two sets as arguments, compares them and returns the difference between the two sets.

The current version of Intel XSLT Accelerator supports all functions of this module defined in the EXSLT resource [6], specifically:

- `set:difference`
- `set:intersection`
- `set:distinct`
- `set:has-same-node`
- `set:leading`
- `set:trailing`
- *Strings* functions are responsible for string manipulation; for example, the `str:tokenize` function splits up a string and returns a node set of token elements, each containing one token from the string.

The current version of Intel XSLT Accelerator supports most functions of this module defined in the EXSLT resource [6], specifically:

- `str:tokenize`
- `str:padding`
- `str:align`
- `str:concat`
- `str:split`

The current version of the library does not implement the following EXSLT functions of this module: `str:replace`, `str:encode-uri`, and `str:decode-uri`.

The current version of Intel XSLT Accelerator does not support the following EXSLT modules: *dynamic*, *functions*, *random*, and *regular expressions*.



---

**NOTE.** Intel XSLT Accelerator does not support alternative language implementations of EXSLT functions. For example, `{http://exslt.org/functions}:func` is not supported, so you cannot make use of implementations in JavaScript.

---

Intel® XSLT Accelerator is fully compatible with the Xalan\* XSLTC processor [5] in extension function support.

## Using EXSLT Functions

For detailed instructions on how to use EXSLT functions, see the EXSLT website [6]. This section tells you how to call an EXSLT function. Follow the steps below.

1. Declare a namespace referring to the EXSLT module that contains the desired function. For example, to call a date-and-time function, type:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:date="http://exslt.org/dates-and-times" >
...
</xsl:stylesheet>
```

2. Call the extension function in your templates. For example, get the year as a number type from the input string, call the `date:year` function in the following way:

```
<xsl:value-of select="date:year()">
```



---

**NOTE.** By default, the extension namespace is output into the result tree. To prevent this, specify the `extension-element-prefixes` attribute value with the module namespace prefix.

---

## User-Defined Extension Functions

User-defined extension functions enable you to augment transformations with custom functions. Extension functions can operate with several XSLT and non-XSLT object types as input arguments and return values. This section describes each object type in turn and lists the Java\* argument types that the object type can map to.

### Input XSLT Types

Intel® XSLT Accelerator supports passing the following types of XSLT objects as arguments to extension functions:

- Simple types: string, boolean or number
- Node-set types

- Result tree fragments as unchangeable sets of nodes

**Table 3-1 Mapping XSLT Object Types to Java\* Argument Types**

<b>XSLT Type</b>	<b>Java* Argument Type</b>
Node-set	org.w3c.dom.traversal.NodeIterator
	org.w3c.dom.Node or its subclasses
	org.w3c.dom.NodeList
	java.lang.String
	java.lang.Object
	char
	[double, float, long, int, short, byte]
	boolean
Result tree fragment	Same as for the Node-set type.
String	java.lang.String
	java.lang.Object
	boolean
	char
	[double, float, long, int, short, byte]
Boolean	boolean
	java.lang.String
	java.lang.Object
	java.lang.Boolean
Number	char
	boolean
	[double, float, long, int, short, byte]
	java.lang.String
	java.lang.Object
	java.lang.Double



**NOTE.** Nodes passed to extension functions as node-sets or result tree fragments are read-only and cannot be modified by the extension function.

## Return Values

Extension functions can return the following value types:

- Simple XSLT types: strings, number types and Boolean values
- XSLT node-sets that correspond to `org.w3c.dom.traversal.NodeIterator` and `org.w3c.dom.Node` Java\* return types
- Result tree fragments to be contributed to the XSLT result document that corresponds to `org.w3c.dom.DocumentFragment` Java type.

## Non-XSLT Types

Non-XSLT types are Java\* objects that are passed as stylesheet parameters or returned by extension functions. Non-XSLT object types are mapped to Java arguments as follows:

- native types or superclasses
- `double`
- `float`
- `long`
- `int`
- `short`
- `char`
- `byte`
- `java.lang.String`

## Namespace Declaration for Extension Functions

To declare the namespace for an extension function, you can use abbreviated syntax; for example, `xmlns:my-class="java.util.Hashtable"`.

In addition to the abridged syntax, Intel® XSLT Accelerator supports Xalan-specific namespace declaration syntax for Java\* extensions. With this abbreviated Unique Resource Identifier (URI) syntax, you can declare the namespace for your extension functions in one of the following formats:

**Table 3-2**      **Formats for Xalan Namespace Declaration**

Name	Format
Class format	<p>xmlns:my-class="xalan://<b>full_class_name</b>"</p> <p>Where <b>full_class_name</b> is the fully qualified class name.</p> <p>Examples</p> <pre>xmlns:my-class="xalan://java.util.Hashtable" xmlns:my-class="xalan://mypackage.myclass"</pre>
Package format	<p>xmlns:my-package="xalan://<b>package_name</b>"</p> <p>Where <b>package_name</b> is the beginning of the Java* package name.</p> <p>Examples</p> <pre>xmlns:my-package="xalan://java.util" xmlns:my-package="xalan://mypackage"</pre> <p>Note that unlike Apache* XSLTC, Intel XSLT Accelerator has no concept of a default object for extension functions.</p>
Java* format	<p>xmlns:java="http://xml.apache.org/xalan/java"</p>

For details on usage patterns for specific formats, please refer to the Apache\* Xalan-Java\* web resource [\[4\]](#).

## Using User-defined Extension Functions

The steps to invoke a user defined extension function are similar to those used to invoke an EXSLT function. This section provides a realistic example to illustrate the calling procedure.

1. Declare the namespace for your function(s), for example:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:my-class="xalan://java.lang.Integer">
```

2. Call the extension function in your templates, for example:

```
<xsl:variable name="new-pop"
select="my-class:valueOf("12345")"/>
```



---

**NOTE.** You can call an extension function inside another extension function both for user-defined and for EXSLT extension functions.

---

## Other Extensions

This part groups any other extensions supported by Intel® XSLT Accelerator for Java\* Environments that are not EXSLT or user-defined extensions.

### Redirect Extension

This is an extension specific for Apache Xalan-Java\* [4] that enables you to redirect parts of the transformation output to one or more files. You can see how this extension is used by running the `RedirectExample` application in the `/IntelXsltJ/Examples` directory. For instructions on running this example, see [“About Examples”](#).

# *Internal Characteristics*

---

# 4

This part defines operational specifics of Intel® XSLT Accelerator for Java\* Environments and gives tips on getting maximum performance with this product.

## Threading Support

Intel® XSLT Accelerator conforms to the threading models that are specified in the JAXP description [3].

## Performance

Intel® XSLT Accelerator combines the convenient JAXP Java\* interface with the effectiveness of a native library. This section contains steps for getting peak performance out of the product.

Intel XSLT Accelerator shows best results in:

- Multithreaded environments
- Stream-to-stream transformations
- Continuous transformation workload

### Benefit from Multithreading

Intel® XSLT Accelerator for Java\* Environments is a thread-safe library. Your Java\* application can use standard Java threads to perform multithreaded transforms.

Intel XSLT Accelerator is built and optimized to be efficiently thread-scalable. This library reaches its peak performance when the number of application threads equals to the number of physical CPUs. For example, for the Intel® Core™2 Duo processor, the peak performance of your application with Intel XSLT Accelerator is for four threads.

When thousands of threads are running, the load of context switching and locking can outweigh the benefits of multithreading. However, even in such a challenging environment, Intel XSLT Accelerator is more stable and significantly better than competition.

## Speed up your XSLT Transformations

Intel® XSLT Accelerator is optimized to efficiently perform stream-to-stream (byte or character based) transformation, such as XML-to-XML and XML-to-HTML transformations.

Intel XSLT Accelerator improves application performance for read and write operations over data on disk or in the network. For an example of stream-to-stream transformation, see `TransformExample` in the `/IntelXsltJ/Examples` directory.

## Manage Java\* Memory Efficiently

Intel® XSLT Accelerator uses native memory pools for XSL processing, which reduces usage of Java\* memory compared to a pure Java solution. As a result, Java memory is less fragmented and JVM\* invokes the garbage collector less often thus freeing CPU for transformation processing.

## Installation Checks

### Error message or problem

```
javax.xml.transform.Transformer  
FactoryConfigurationException:  
Provider  
com.intel.xml.transform.Transfo  
rmerFactoryImpl could not be  
instantiated
```

```
Exception in thread "main"  
java.lang.UnsatisfiedLinkError:  
no intel-xslt in  
java.library.path  
Failed to launch the library.
```

### Suggested fix

Check that the file `intel-xslt-1_1_0.jar` is present in the `CLASSPATH` value; see section [“Configuring the Environment”](#).

Check that the path to `intel-xslt.dll` (on Windows\*) or `libintel-xslt.so` (on Linux\*) is listed in the `java.library.path` property of the JVM\*.

- Check that you have J2SE\* of version 1.5.0.
- Verify that your environment variables are configured to point to the correct locations. For details on setting the required variables, see the section [“Configuring the Environment”](#). You can confirm the values for the path variables on the command line, for example:

On Linux\*:

```
echo $LD_LIBRARY_PATH
```

```
echo $CLASSPATH
```

On Windows\*:

```
echo %PATH%
```

```
echo %CLASSPATH%
```

## Running the examples

### Error message or problem

During compilation of an example from the Apache\* Xalan-Java\* website, class `org.apache.xml.serializer.Serializer` is not found.

During execution of the example, the following error is produced:

```
java.lang.NoClassDefFoundError:  
org/apache/xml/serializer/...
```

### Suggested fix

Make sure that the provided `serializer.jar` file is in the value of the `CLASSPATH` variable.

Make sure that the provided `serializer.jar` file is in the value of the `CLASSPATH` variable.

# Index

---

## A

Accelerator, 1-4  
    characteristics  
        thread support, 4-1  
    configuring, 2-4  
    installing, 2-1  
    performance, 4-1  
    quick start, 2-5  
    switching, 2-5  
    using, 3-1

## C

class format, 3-11

## D

document object model, 3-4  
DOM (document object model), 3-4

## E

EXSLT extensions, 3-4, 3-5  
    Common, 3-5  
    Date-and-time, 3-5  
    Math, 3-6  
    Sets, 3-7  
    String, 3-7  
    using, 3-8  
extensible stylesheet language, 1-4  
    transformation, 1-4  
extensions  
    EXSLT See EXSLT extensions  
    user-defined See user-defined extensions

## I

Intel® XSLT Accelerator, 1-4

## J

Java API for XML Processing, 1-7  
Java format, 3-11  
Java Native Interface, 1-8  
JAXP (Java API for XML processing), 1-7  
JNI (Java Native Interface), 1-8

## N

namespace  
    declaration format of, 3-10  
    declaring, 3-8, 3-10  
node-set, 3-9

## O

objects, 3-4  
    DOM trees, 3-4  
    SAX Events, 3-4  
    streams, 3-4

## P

package format, 3-11  
processor  
    Xalan XSLTC, 1-3, 1-6  
    Xalan-J, 1-3, 1-6

## R

result tree fragment, 3-9

## S

SAX (Simple API for XML), 3-4

Simple API for XML, 3-4

## T

transformation

- API, 1-7

- configuring, 3-3

- identity, 1-4

- object types, 3-4

- performing, 3-1

- XSL, 1-4

transformer, 3-2

- structure, 1-8

  - Java XML, 1-8

  - native core, 1-8

- type

  - DocumentBuilder, 3-4

  - SAXParser, 3-4

transformer factory, 3-1

## U

user-defined extensions

- non-XSL input types, 3-10

- return values, 3-10

- using, 3-11

- XSL input types, 3-8

## X

Xalan

- namespace declaration, 3-11

XML, 1-4

- processors, 1-4

XSL

- extensions, 3-4

XSL (extensible stylesheet transformation), 1-4