

Release Notes

XMLMill for Java - XMLMill for Domino

2.20

xmlmill

© 2000-2005 Pecunia Data Systems bvba. All rights reserved.

NOTICE: All information contained herein is the property of Pecunia Data Systems bvba.

This publication and the information herein are furnished AS IS, are subject to change without notice, and should not be construed as a commitment by Pecunia Data Systems bvba. Pecunia Data Systems bvba assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third-party rights.

Table of Contents

1. Preface	2
2. Upgrading from previous versions	3
2.1. Upgrading from version 2.10	3
2.2. Upgrading from earlier versions	3
3. What's new ?	4
3.1. XMLMill for Java	4
3.2. XMLMill for Domino	4
4. XMLMill for Java	5
4.1. JAXP Compliancy	5
4.1.1. Getting a JAXP 1.3 implementation	5
4.2. XMLMill Configuration	5
4.2.1. Element: <jaxp>	5
4.2.2. Element: <implementation>	5
4.2.3. Element: <factories>	6
4.2.3.1. Element: <saxparser-factory>	6
4.2.3.2. Element: <document-builder-factory>	6
4.2.3.3. Element: <transformer-factory>	6
4.2.3.4. Element: <schema-factory>	6
4.3. XMLMill Interactive	6
4.3.1. Menu options	7
4.3.1.1. Configuration	7
4.4. XMLMill Batch	7
4.4.1. GNU Long Style Options	7
4.4.2. New arguments	8
4.5. XMLMill in java	8
4.5.1. Class: JAXPTransformer	9
4.5.1.1. Removed methods	9
4.5.1.2. Deprecated methods	9
4.5.1.3. New methods	9
4.5.1.3.1. void setBaseURL()	9
4.5.1.3.2. void setTemplates()	10
4.5.1.3.3. void setSchema()	10
4.5.1.3.4. void setSchemaLocation()	10
4.5.1.4. Example: TestAppl.	10
4.5.2. Class: Configurator	11
4.6. XMLMill and JS2E	12
4.6.1. XMLMill / JS2E 1.3	12
4.6.2. XMLMill / JS2E 1.4	12
4.6.3. XMLMill / JS2E 1.5	13
4.6.4. XMLMill / Saxon 8.0 / JS2E 1.4	13
4.6.5. XMLMill / Saxon 8.0 / JS2E 1.5	13
5. XMLMill for Domino	15
5.1. New functionality	15
5.1.1. Query_String in Agent	15
5.1.2. Query_String in XSL Document	15
5.2. Changes in servlets.properties	15
5.2.1. Adding a logfile parameter	15

6. New attributes.17
6.1. Attribute: absolute-position. 17
6.2. Attribute: relative-position.20
6.3. Attribute: wrap-option.21

7. Bugfixes.22



1. Preface

The XMLMill application is intended for software developers who want to generate .pdf documents from xml and/or xsl.

For an overview of how to use XMLMill in general please consult in the **docs/** directory:.

- ◆ **apidoc/** -- The JavaDoc concerning the **XMLMill** public api.
- ◆ **userguide.pdf** -- XMLMill user's guide.
- ◆ **dtdguide.pdf** -- An explanation of XMLMill's elements (tags) and their attributes.
- ◆ **samplesguide.pdf** -- An overview of the examples in the **samples/** directory contained in the download.

If you have questions, please do not hesitate to send a mail to support@xmlmill.com.

☞ *This document is completely generated with XMLMill 2.20 using **rnotes220.xml** and **rnotes.xsl**. These files can be found in the directory **samples/docs/rnotes** in the download.*

xmlmill

2. Upgrading from previous versions

2.1. Upgrading from version 2.10

No changes on .xsl/.mill/.mill or Java applications are required, but the use of the xmlmill.xsd of this version is required in order to be able to use the new functionalities.

2.2. Upgrading from earlier versions

The best procedure to make your transition as smooth as possible is as follows:

1. Back up all your xml/xsl/mill files or Java code.
2. Read all release notes <http://www.xmlmill.com/news.html> from your current version to the latest version in correct order (from oldest release notes to newest).
3. Modify your xml/xsl/mill files or Java code accordingly to the release notes.

xmlmill

3. What's new ?

3.1. XMLMill for Java

This release notes document describes all new (or enhanced) functionalities of version 2.20 of XMLMill. The main changes are:

- ◆ As of this version XMLMill is JAXP (1.2 or 1.3) compliant, allowing the use of a vendor specific XML parser and XSL transformer (which are JAXP compliant).
- ◆ The Command Line Interface (CLI) has been revised to include the GNU-style long option format.
- ◆ A limited list of attributes are added.
- ◆ Bug fixes

3.2. XMLMill for Domino

This release notes document describes all new (or enhanced) functionalities of version 2.20 of XMLMill. The main changes are:

- ◆ The temporary document contains an extra field called **Query_String** containing the query information appended to the URL following the question mark (so all parameters passed to XMLMill).
- ◆ All parameters defined in the **Query_String** are also available as global stylesheet parameters.
- ◆ A limited list of attributes are added.

xmlmill

4. XMLMill for Java

4.1. JAXP Compliancy

The biggest modification made to this version of XMLMill is the it has become JAXP compliant (1.2 or 1.3).

JAXP allows you to use any XML-compliant parser from within your application. It does this with what is called a pluggability layer, which allows you to plug in an implementation of the SAX or DOM APIs. The pluggability layer also allows you to plug in an XSL processor, letting you control how your XML data is transformed.

☞ For more information regarding the latest version of JAXP (1.3) please visit the JAXP home page at <http://java.sun.com/xml/jaxp/>

4.1.1. Getting a JAXP 1.3 implementation

The latest version of the Java API for XML Processing (JAXP1.3) is now final and part of J2SE 5.0. The implementation is also available as a separate download at <http://jaxp.dev.java.net/> for developers to experiment in previous versions of J2SE (1.3 and 1.4).

Following **jars** should be added to the classpath (and replacing any previous .jars representing an xml-parser and xsl-transformer) :

- ◆ xercesImpl.jar
- ◆ xalan.jar
- ◆ dom.jar
- ◆ jaxp-api.jar
- ◆ sax.jar

☞ Apache's current Xalan-J version (2.6) (see <http://xml.apache.org/xalan-j/> for more information) is JAXP 1.2 compliant.

4.2. XMLMill Configuration

The built-in configuration file (**config.xml** located in the **com\xmlmill\conf** directory in the **xmlmill.jar** file) has been modified to reflect the JAXP compliancy. Following elements have been added:

4.2.1. Element: **<jaxp>**

The children nodes of this element define the implementation version of the JAXP specification and the actual implementation classes of the different JAXP factories (see below).

The **<jaxp>** element has following children:

```
<jaxp>
  <implementation version="..." />
  <factories>
    <saxparser-factory      implementation-class = "..." />
    <document-builder-factory implementation-class = "..." />
    <transformer-factory    implementation-class = "..." />
    <schema-factory        implementation-class = "..." />
  </factories>
</jaxp>>
```

4.2.2. Element: **<implementation>**

This element defines the underlying JAXP implementation version. The default **version** is **1.2** to stay compatible with previous versions of XMLMill.

The allowed values are 1.2 and 1.3

☞ Remember that validation of the result-tree can only be done with a JAXP 1.3 implementation.

4.2.3. Element: <factories>

This element defines the underlying implementation classes of following factories (abstract classes):

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

The <factories> element has following children:

```
<factories>
  <saxparser-factory      implementation-class = "..."/>
  <document-builder-factory implementation-class = "..."/>
  <transformer-factory   implementation-class = "..."/>
  <schema-factory        implementation-class = "..."/>
</factories>
```

4.2.3.1. Element: <saxparser-factory>

This element has exactly one attribute: **implementation-class** defining the name of the class in the underlying implementation that will be used to create an instance of this factory.

For example: If you use JDK1.5 the SAXParser factory is implemented in following class:
com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl

☞ Please check your vendor's documentation regarding the exact class name to define.

4.2.3.2. Element: <document-builder-factory>

This element has exactly one attribute: **implementation-class** defining the name of the class in the underlying implementation that will be used to create an instance of this factory.

For example: If you use JDK1.5 the Document Builder factory is implemented in following class:
com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl

☞ Please check your vendor's documentation regarding the exact class name to define.

4.2.3.3. Element: <transformer-factory>

This element has exactly one attribute: **implementation-class** defining the name of the class in the underlying implementation that will be used to create an instance of this factory.

For example: If you use JDK1.5 the Transformer factory is implemented in following class:
org.apache.xalan.processor.TransformerFactoryImpl

☞ Please check your vendor's documentation regarding the exact class name to define.

4.2.3.4. Element: <schema-factory>

This element has exactly one attribute: **implementation-class** defining the name of the class in the underlying implementation that will be used to create an instance of this factory.

For example: If you use JDK1.5 the Transformer factory is implemented in following class:
com.sun.org.apache.xerces.internal.jaxp.validation.xs.SchemaFactoryImpl

☞ Please check your vendor's documentation regarding the exact class name to define.

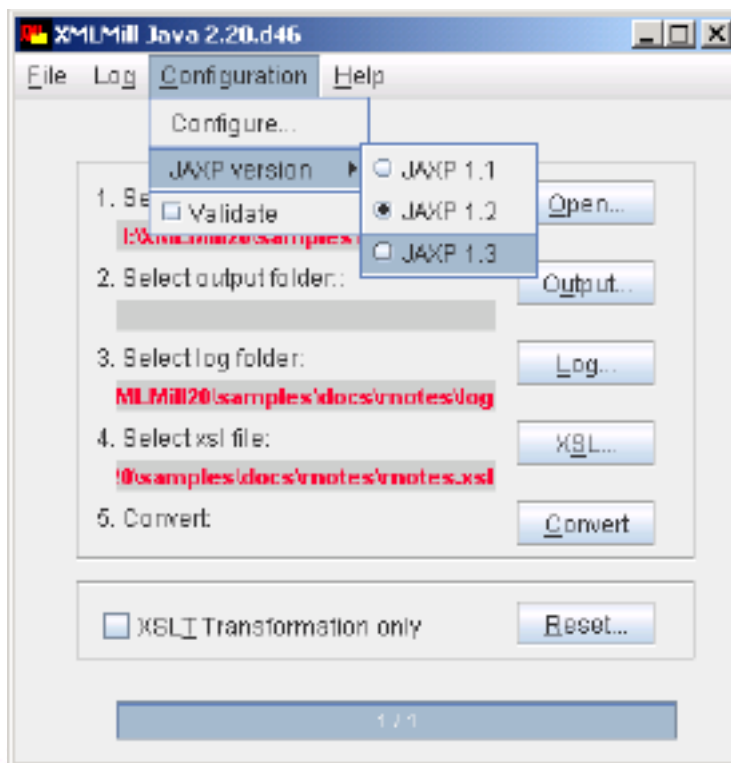
4.3. XMLMill Interactive

XMLMill InterActive has been modified in order to reflect XMLMill's JAXP compliancy.

4.3.1. Menu options

4.3.1.1. Configuration

The configuration menu has been extended as follows:.



A menu-option **JAXP version** has been added so the underlying implementation version can be selected.

A checkbox **Validate** has been added indicating that the result-tree should be validated against the **xmlmill** schema.

☞ *The validation will always be done against the **xmlmill.xsd** defined in the **com/xmlmill/resources/xsd** directory in the **xmlmill.jar** file.*

4.4. XMLMill Batch

This version extends the command-line interface of XMLMill with the GNU-style long option format. This format consist of two dashes and a keyword, facilitating defining the arguments to pass.

4.4.1. GNU Long Style Options

Please find below the original POSIX options and corresponding GNU-style long options:

POSIX style	GNU Long style	Description
-?	--help	Prints the on-line help message.
-a	--append	Append messages to an existing log file instead of overwriting the log file. If the log file does not exist, it is created.
-c	--config	Defines the configuration file to use.
-e	--errorflags	Defines on which errors XMLMill should abort

POSIX style	GNU Long style	Description
	<code><integer></code>	generation of the pdf document. This parameter is the sum (combination) of following values: (2) abort on errors (8) abort on warnings. Note that a fatalError will always be logged and will lead to aborting the generation.
<code>-g</code>	<code>--loglevel <level></code>	Defines the loglevel. The values can be: DEBUG, INFO, WARN, ERROR, FATAL, ALL, OFF. If the value passed is different than the ones defined here, the loglevel is set to DEBUG.
<code>-l</code>	<code>--log <path/file></code>	Defines an (existing) logfolder and/or logfile name. If a file-name is defined it should end with .log. If the log-folder only contains a filename (ending in .log) the log will be written in directory of first xml-file that is processed. If no log-folder is defined, no log-folder will be written. All log messages (if any) will be sent to the screen.
<code>-m</code>	<code>--xml <path/file></code>	Defines the .xml or .mill file or directory containing .xml or .mill files. Multiple -m files can be defined.
<code>-o</code>	<code>--outputfolder <directory></code>	Defines the (existing) outputfolder. If no outputfolder is defined, the output will be written in the directory defined by file attribute of the <code><output></code> tag of the first xml-file processed.
<code>-s</code>	<code>--xsl <file></code>	Defines the .xsl file that will be used to transform the .xml files. If specified this .xsl file will overrule any .xsl file defined in the .xml file (defined in the <code><?xml-stylesheet ...?></code> tag).
<code>-t</code>	<code>--xslt</code>	XSLT transformation only (transform .xml/.xsl file to a .mill file).
<code>-v</code>	<code>--verbose</code>	Enable verbose output
<code>-x</code>	<code>--size <integer></code>	Define the minimum size (in bytes) the pdf document should be.

☞ The validation will always be done against the `xmlmill.xsd` defined in the `com/xmlmill/resources/xsd` directory in the `xmlmill.jar` file.

☞ None of POSIX options and arguments are changed.

4.4.2. New arguments

Due to the added JAXP compliancy of this version following arguments have been added:

POSIX style	GNU Long style	Description
<code>-i</code>	<code>--validate <level></code>	Validates the result tree. Note: This requires a JAXP 1.3 compliant implementation of the underlying transformer.
<code>-j</code>	<code>--jaxp <JAXP version></code>	Define the JAXP version of the underlying implementation (1.2 or 1.3) (default: 1.2).

4.5. XMLMill in .java

In order to reflect JAXP compliancy the **PDXTransform** class has become deprecated and has been replaced by the **JAXPTransformer** class. Following changes were done:

4.5.1. Class: JAXPTransformer

4.5.1.1. Removed methods

Following methods were already deprecated in the **PDXTransform** class and not withheld in the **JAXPTransformer** class.

Removed PDXTransform method	New JAXPTransformer method
setXSLTTransformationOnly(boolean)	Use setFeature(String) instead.
printComments(boolean)	Use setFeature(String) instead.

4.5.1.2. Deprecated methods

Following methods are deprecated and replaced by equivalent methods (for more consistency):

Deprecated PDXTransform method	New JAXPTransformer method
setXMLFile(File)	setXMLDocument(File)
setXMLStream(Stream)	setXMLDocument(Stream)
setXMLReader(Reader)	setXMLDocument(Reader)
setXMLUrl(URL)	setXMLDocument(URL)

Deprecated PDXTransform method	New JAXPTransformer method
setXSLFile(File)	setXSLDocument(File)
setXSLStream(Stream)	setXSLDocument(Stream)
setXSLReader(Reader)	setXSLDocument(Reader)
setXSLUrl(URL)	setXSLDocument(URL)

Deprecated PDXTransform method	New JAXPTransformer method
setXMLSystemId(String)	setBaseUrl(String)
setXMLSystemId(File)	setBaseUrl(File)
setXMLSystemId(URL)	setBaseUrl(URL)
setXSLSystemId(String)	setBaseUrl(String)
setXSLSystemId(File)	setBaseUrl(File)
setXSLSystemId(URL)	setBaseUrl(URL)

☞ For more information please visit the JavaDoc API contained in the download or online at <http://www.xmlmill.com/products/docs/apidocs/index.html>.

4.5.1.3. New methods

Following methods have been added (the corresponding **get** methods are omitted for brevity):

4.5.1.3.1. void setBaseUrl()

This method replaces the previous (unclear) **setXSLSystemId()** and **setXMLSystemId()**

methods.

It defines the URL against which relative URLs in the result-tree will be resolved. Generally, it's only necessary to set this property if the result-tree contains relative URLs.

4.5.1.3.2. void setTemplates()

Defines a **javax.xml.transform.Templates** instance used to define an internal **javax.xml.transform.Transformer** instance. If a **Templates** instance is defined, it will take precedence over a xsl-styleSheet defined in an xml document (using the `<?xml-styleSheet href="..." type="text/xsl"?>` tag).

☞ You need a JAXP 1.3 compliant implementation to use this method successfully

4.5.1.3.3. void setSchema()

Defines a **javax.xml.validation.Schema** instance. The **javax.xml.validation.Schema** class is an "immutable" memory representation of schema. Once the schemas are loaded into memory and represented as a **Schema** instance, it can be used for validation. In this way, a single **Schema** instance can be shared with many different parser instances even running in different threads. Applications are encouraged to be written so that the same set of schema are parsed only once and the same **Schema** instance is passed to different instances of the parser.

☞ You need a JAXP 1.3 compliant implementation to use this method successfully

4.5.1.3.4. void setSchemaLocation()

Defines a **javax.xml.validation.Schema** to use to validate the result-tree against

☞ You need a JAXP 1.3 compliant implementation to use this method successfully

☞ For more information please visit the JavaDoc API contained in the download or online at <http://www.xmlmill.com/products/docs/apidocs/index.html>.

4.5.1.4. Example: TestAppl

The following class defines the xml file and xsl file as **File** objects and a **JAXPTransformer** instance generates the .pdf document.

```
package test.xmlmill;

import com.xmlmill.*; ❶
import java.io.*;
import com.xmlmill.config.Configurator;

public class testAppl extends Object
{
    public static void main(String[] args)
    {
        try {
            // Set Log-level (optional)
            PDXLogger.setLevel("DEBUG"); ❷

            // Set the logfile (optional) ❸
            PDXLogger.setPDXLogFile("file:/C:/XMLMill/samples/xmlxsl/log/phonelist.log",
            true);

            // Define a configurator (optional)
            Configurator.configure(); ❹

            // Define the xml(mill) file to transform ❺
            File xmlfile = new File("C:\\XMLMill\\samples\\xmlxsl\\phonelist.xml");

            // Define a XSL file (optional)
            File xslfile = new File("C:\\XMLMill\\samples\\xmlxsl\\phonelist.xsl"); ❻

            // Define an errorHandler(optional) ❼
            PDXTransformErrorHandler ehandler = new
            PDXTransformErrorHandler(PDXTransformErrorHandler.ABORT_ON_ERROR );
```

```

// Define a transform object ❸
JAXPTransformer transform = new JAXPTransformer();

// Set the xml file ❹
transform.setXMLDocument(xmlfile);

// Set the xsl file ❺
transform.setXSLDocument(xslfile);

// Define the baseURL (used to build a path to relative
// references in the xml or xsl document
// (like src="image.jpg") ❻
transform.setBaseURL(xslfile);

// Set the errorhandler ❼
transform.setErrorHandler(ehandler);

// Let the transform instance know the
// JAXP implementation is 1.3
transform.setVersion("1.3"); ❸❹

// Turn on validation
transform.setValidating(true); ❶❷

// ... and transform the object to a pdf file
transform .transform(); ❶❸
System.out.println("-- Done --");

} catch (Throwable e) {
System.out.println(e.toString());
} finally {
PDXLogFile.resetErrorLog(); ❶❹
transform = null;
}
}

```

- ❶ Be sure to import the **com.xmlmill** package and the **com.xmlmill.Configurator** package.
- ❷ Define the log-level.
- ❸ Define where to send the log messages to.
- ❹ Defines the configurator to use. In case you do not add a parameter that represents an external configuration file, the default internal configuration file will be used.
- ❺ Define a file object referencing the xml(mill) file that needs to be converted to a PDF file.
- ❻ Define a file object referencing the xsl file that will perform the transformation. This is optional if the xml file contains a reference to a xsl file with the **<?xml-stylesheet>** tag.
- ❼ Define an errorhandler that will handle all **warnings, errors, fatalError**s.
- ❸ Define a JAXP compliant transform object.
- ❹ Define the xmlfile to be processed.
- ❺ Define the xslfile to be used.
- ❶❶ Define the baseURL used to define the location of relative references in the xml or xsl document.
- ❶❷ Set the errorhandler.
- ❶❸ Let the transform instance know the JAXP implementation is 1.3.
- ❶❹ Turn on validation.
- ❶❸ Do the transformation.
- ❶❹ Finally reset the error log and dereference any variables.

The result will be written in a file defined by the **file** attribute in the **<ml:document>** tag. You can override this behavior by using the **setOutput(java.io.OutputStream)** method, allowing you to send the data to a stream or to another subsystem.

4.5.2. Class: **Configurator**

The configurator's `configure()` method has been extended so the JAXP version can be defined when XMLMill is configured.

Eight overloaded `configure()` methods are available:

Method	JAXP Version implemented
<code>configure()</code>	Loads the configuration values from the default config.xml file with the JAXP version as defined in the default config.xml file.
<code>configure(int i)</code>	Loads the configuration values from the default config.xml file with the JAXP version passed as parameter. The allowed parameters are: Configure.JAXP_12 , Configure.JAXP_13
<code>configure(File f)</code> <code>configure(String s)</code> <code>configure(URL url)</code>	Loads the configuration values from the specified config.xml file with the JAXP version as defined in that config.xml file.
<code>configure(File f, int i)</code> <code>configure(String s, int i)</code> <code>configure(URL url, int i)</code>	Loads the configuration values from the specified config.xml file with the JAXP version passed as parameter. The allowed parameters are: Configure.JAXP_12 , Configure.JAXP_13

☞ *If the JAXP version is passed as parameter it takes precedence over the value defined in the config.xml file.*

☞ *The JAXP version is defined at the moment XMLMill is configured and will stay valid for the duration of the JVM session.*

4.6. XMLMill and JS2E

In the following sections an overview is given of how to use XMLMill in different environments. As XMLMill is JAXP compliant you should be able to use XMLMill in any environment that supports JAXP.

4.6.1. XMLMill / JS2E 1.3

As the JDK 1.3 version does not include any xml parser nor xsl transformer libraries, you need to add external libraries to the classpath, for example:

```
set JAVA_HOME= "C:\Program Files\JavaSoft\sdk131_07\bin"
set XML_HOME=c:\xmlmill2.20

set CLASSPATH=%XML_HOME%\lib\xercesimpl.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xalan.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\jaxp-api.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\sax.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\dom.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xmlmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\log4j.jar

%JAVA_HOME%\java com.xmlmill.applet.Main
```

☞ *See the `etc/batch.cmd` or `etc/gui.cmd` command file for an example*

4.6.2. XMLMill / JS2E 1.4

As the JDK 1.4 version does include an xml parser and xsl-transformer, there is **no need** to add external libraries to the classpath, for example:

```
set JAVA_HOME= C:\j2sdk1.4.2_04\bin
set XML_HOME=c:\xmlmill2.20

set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xlmmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\log4j.jar

%JAVA_HOME%\java com.xmlmill.applet.Main
```

☞ See the **etc/batch.cmd** or **etc/gui.cmd** command file for an example

4.6.3. XMLMill / JS2E 1.5

As the JDK 1.5 version does include an xml parser and xsl-transformer, there is **no need** to add external libraries to the classpath, for example:

```
set JAVA_HOME= "C:\Program Files\Java\jdk1.5.0_01\bin"
set XML_HOME=c:\xmlmill2.20
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-jdk5.0\xlmmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-jdk5.0\log4j.jar

%JAVA_HOME%\java com.xmlmill.applet.Main
```

☞ See the **etc/batch.cmd** or **etc/gui.cmd** command file for an example

4.6.4. XMLMill / Saxon 8.0 / JS2E 1.4

Saxon (<http://www.saxonica.com/>) is an XSLT transformer which can be used as an alternative of Xalan. To use Saxon, you need to add the **dom.jar** and **jaxp-api.jar** libraries of the JAXP 1.3 implementation (<http://java.sun.com/xml/jaxp/>) to the classpath in order to be able to do validation. Adding a xml-parser is not necessary as the default (Xerces) parser included in the SDK1.4 will be used:

```
set JAVA_HOME= C:\j2sdk1.4.2_04\jre\bin

set XML_HOME=c:\xmlmill2.20
set CLASSPATH=%XML_HOME%\lib-saxon8.2b-jdk1.4\saxon8.2a.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\dom.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\jaxp-api.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\xlmmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\log4j.jar

%JAVA_HOME%\java -Djaxp.debug=true
-Djavax.xml.validation.SchemaFactory:http://www.w3.org/2001/XMLSchema=net.sf.saxon.TransformerFactoryImpl
-Djavax.xml.transform.TransformerFactory=net.sf.saxon.TransformerFactoryImpl
com.xmlmill.applet.Main
```

☞ Do not forget to add the **-D** property indicating that the Saxon XSL transformer and Validator should be used instead of the transformer (Xalan) defined in the JS2E 1.4 version.

☞ See the **etc/batch.cmd** or **etc/gui.cmd** command file for an example.

☞ If you add the **-Djaxp.debug=true** option more information will be printed regarding the selection of the implementation classes (the information will be printed directly on the console).

4.6.5. XMLMill / Saxon 8.0 / JS2E 1.5

Saxon (<http://www.saxonica.com/>) is an XSLT transformer which can be used as an alternative of Xalan. To use Saxon with JS2E 1.5, the **saxon.jar** should be added to the classpath, for example:

```
set JAVA_HOME= C:\j2sdk1.4.2_04\jre\bin
```

```
set XML_HOME=c:\xmlmill2.20
set CLASSPATH=%XML_HOME%\lib-saxon8.2b-jdk1.4\saxon8.2a.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\xmlmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib-saxon8.2b-jdk1.4\log4j.jar

%JAVA_HOME%\java -Djaxp.debug=true
-Djavax.xml.validation.SchemaFactory:http://www.w3.org/2001/XMLSchema=net.sf.saxon.TransformerFactoryImpl
-Djavax.xml.transform.TransformerFactory=net.sf.saxon.TransformerFactoryImpl
com.xmlmill.applet.Main
```

- ☞ *Do not forget to add the **-D** property indicating that the Saxon XSL transformer and Validator should be used instead of the transformer (Xalan) defined in the JS2E 1.5 version.*
- ☞ *See the **etc/batch.cmd** or **etc/gui.cmd** command file for an example*

xmlmill

5. XMLMill for Domino

5.1. New functionality

5.1.1. Query String in Agent

As of this version it is possible to pass **Query_String** parameters to your Lotuscript agent. This way you can extend the functionality of your application.

Use following approach:

1. Behind a **Submit** button you add the parameters to be passed to your agent (using the Formula language).
2. When the form is submitted the parameters are passed to XMLMill.
3. In the temporary document a field called **Query_String** is added containing the query information appended to the URL following the question mark (so all parameters passed to XMLMill).
4. In your script you can to read and parse the **Query_String** field of the temporary document.

5.1.2. Query String in XSL Document

As of this version all **Query_String** parameters are available as global stylesheet parameters in an XSL Document.

Use following approach:

1. Behind a **Submit** button add the parameters to be passed to the XSL Document (using the Formula language).
2. When the form is submitted the parameters are passed to XMLMill.
3. XMLMill passes the parameters to the XSL Document
4. In order to use the parameter(s) in the .xsl file, the .xsl needs to have defined corresponding top-level (global) parameters.

```
<?xml version="1.0" encoding="UTF-16" ?>
<xsl:stylesheet version="1.0"
  xmlns:ml="http://www.xmlmill.com/XSL/Format"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.xmlmill.com/XSL/Format ../../docs/xsd/xmlmill.xsd"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>

<xsl:output method="xml" indent="yes" encoding="UTF-16" />
<xsl:param name="sortcolumn"/> ❶
<xsl:param name="sortorder"/>
...
```

- ❶ Top-level (global) parameters are defined in the .xsl stylesheet with names corresponding to the parameter names defined in the **Query_String**.

As the parameters are global they will be visible in any template in the .xsl stylesheet.

5.2. Changes in `servlets.properties`

5.2.1. Adding a **logfile** parameter

It is possible to define the location of a log-file where XMLMill's messages should be sent to (instead of the console):

You need to define the location of the logfile in your servlet.properties file, so you should add an extra parameter to the servlet.properties file. In this case the servlet.properties file could look like this (pay attention to the logfile-parameter in the example below):

```
# Defines an alias name for the com.xmlmill.domino.xmlmillr6 servlet.
servlet.xmlmillr6.code=com.xmlmill.domino.xmlmillr6

# Define the database, username and password. Make user there are no spaces in the
username.
servlet.xmlmillr6.initArgs= db1=usecases.nsf xmlmill xmlmill, loglevel=INFO ,
logfile=file:/d:/lotus/domino/data/log/xmlmill.txt

# Define that the servlet is initialised when starting the Domino server.
servlets.startup = xmlmillr6
```

☞ *The value should represent the location and name of the logfile as an URI. The URI should be fully resolved.*

xmlmill

6. New attributes

This release has also some new attributes added to some elements.

☞ *For a full overview of all tags and attributes, please visit the [dtdGuide.pdf](#) document in the docs/ directory in the download.*

6.1. Attribute: absolute-position

Value: auto | absolute | fixed

Initial: auto

Applies to: ml:barcode | ml:external-graphic | ml:leader | ml:textbox

Inherited: no

Description:

The **absolute-position** attribute defines if the element is taken out of the normal flow. This means they have no impact on the layout/positioning of other elements displayed in the region. Also, though absolutely positioned areas have margins, they do not collapse with any other margins.

Values have the following meaning:

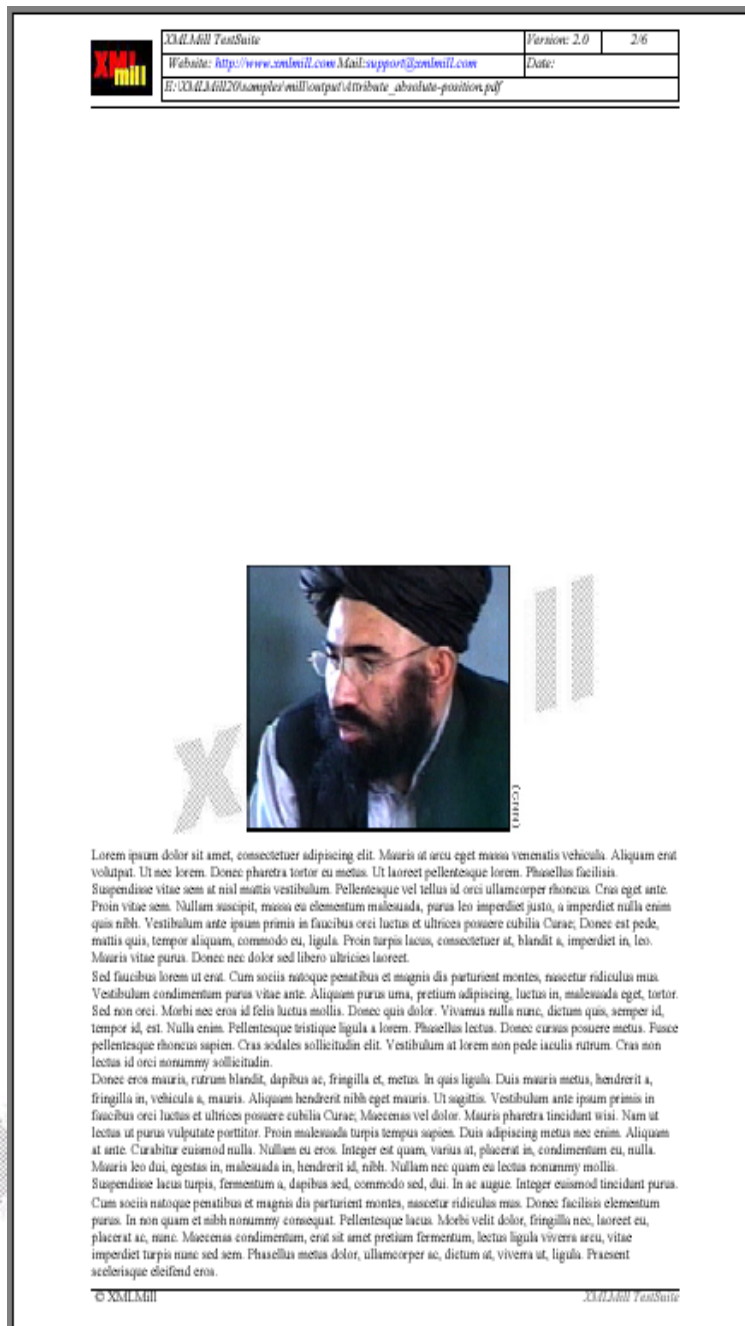
- auto** There is no absolute-positioning constraint. The element is placed in the normal flow of elements being placed in the region.
- absolute** Absolutely positioned elements are taken out of the normal flow. This means they have no impact on the layout/positioning of other elements displayed in the region. Also, though absolutely positioned areas have margins, they do not collapse with any other margins.
- fixed** Absolutely positioned elements are taken out of the normal flow. This means they have no impact on the layout/positioning of other elements displayed in the region. Also, though absolutely positioned areas have margins, they do not collapse with any other margins.

Example 1:

Following example will display an image centered on the page and the text will go over the image (as the image is taken out of the normal flow).

```
<ml:external-graphic src="file:images/zaeef.jpg"
  absolute-position = "fixed"
  topleft = "center,center"
  content-height = "100%"
  content-width = "100%"
/>
```

The result is :



Example 3:

Following example will display an image centered on the page and the text starts above the the image, as the **text-wrapping** attribute is 'on' and the **absolute-position** attribute is **auto** (so the image is placed in the normal flow of the elements printed in the region).

```
<ml:external-graphic src="file:images/zaeef.jpg"
  absolute-position = "auto"
  text-wrapping = "on"
  topleft = "center,center"
  content-height = "100%"
  content-width = "100%"
/>
```

The result is :

XMLMill TestSuite	Version: 2.0	4/6
Website: http://www.xmlmill.com Mail: support@xmlmill.com	Date:	
E:\XMLMill20\sample\mill\output\attribute_absolute-position.pdf		

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris at arcu eget massa venenatis vehicula. Aliquam erat volutpat. Ut nec lorem. Donec pharetra tortor eu metus. Ut laoreet pellentesque lorem. Phasellus facilisis. Suspendisse vitae sem at nisl mollis vestibulum. Pellentesque vel tellus id orci ullamcorper rhoncus. Cras eget ante. Proin vitae sem. Nullam suscipit, massa eu elementum malesuada, purus leo imperdiet justo, a imperdiet nulla enim quis nibh. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec est pede, mattis quis, tempor aliquam, commodo eu, ligula. Proin turpis lacus, consectetur at, blandit a, imperdiet in, leo. Mauris vitae purus. Donec nec dolor sed libero ultrices laoreet. Sed faucibus lorem ut erat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vestibulum condimentum purus vitae ante. Aliquam purus urna, pretium adipiscing, luctus in, malesuada eget, tortor. Sed non orci. Morbi nec eros id felis luctus mollis. Donec quis dolor. Vivamus nulla nunc, dictum quis, semper id, tempor id, est. Nulla enim. Pellentesque tristique ligula a lorem. Phasellus lectus. Donec curvas posuere metus. Fusce pellentesque rhoncus sapien. Cras sodales sollicitudin elit. Vestibulum at lorem non pede iaculis rutrum. Cras non lectus id orci nonummy sollicitudin.

Donec eros mauris, rutrum blandit, dapibus ac, fringilla et, metus. In quis ligula. Duis mauris metus, hendrerit a, fringilla in, vehicula a, mauris. Aliquam hendrerit nibh eget mauris. Ut sagittis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Maecenas vel dolor. Mauris pharetra tincidunt wisi. Nam at lectus ut purus vulputate porttitor. Proin malesuada turpis tempus sapien. Duis adipiscing metus nec enim. Aliquam at ante. Curabitur euismod nulla. Nullam eu eros. Integer est quam, varius at, placerat in, condimentum eu, nulla. Mauris leo duis, egestas in, malesuada in, hendrerit id, nibh. Nullam nec quam eu lectus nonummy mollis. Suspendisse lacus turpis, fermentum a, dapibus sed, commodo sed, duis. In ac augue. Integer euismod tincidunt purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec facilisis elementum purus. In non quam et nibh nonummy consequat. Pellentesque lacus. Morbi velit dolor, fringilla nec, laoreet eu, placerat ac, nunc. Maecenas condimentum, erat sit amet pretium fermentum, lectus ligula viverra arcu, vitae imperdiet turpis nunc sed sem. Phasellus metus dolor, ullamcorper ac, dictum at, viverra ut, ligula. Praesent acelerisque eleifend eros.

Nam interdum bibendum amet lacus condimentum vestibulum in, iaculis ea, pulvinar faucibus lorem. Nam ullamcorper nec, nonummy sapien sed lacus. Aliquam congue nunc vel duis. Aliquam lacus tristique lobortis. In felis. Proin id neque. Fringilla libero. Aenean nec habitasse platea dictumst. eros. Nulla augue augue. bibendum a, dolor. Proin placerat, wisi tellus tincidunt neque et wisi. Integer ut tortor eros. Quisque nibh. Mauris Maecenas diam ligula, tincidunt ut, tristique at, elementum feugiat, quam. Integer non elit et ligula ullamcorper mollis. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam eleifend, felis consequat aliquet ultrices, metus odio vehicula lorem, ac viverra lacus dolor nec felis. Nullam aliquet nunc elementum elit.

Curabitur wisi mauris, tincidunt quis, malesuada quis, luctus at, nisi. Aliquam id purus vel massa fringilla cursus. Mauris magna elit, sollicitudin vitae, laoreet nec, volutpat nec, neque. Phasellus eleifend semper elit. Vivamus porta arcu sit amet ante pellentesque elementum. Integer odio turpis, eleifend vitae, placerat id, placerat eget, tellus. Nam ligula dolor, tempor ut, tincidunt eget, placerat quis, tortor. In ultrices ultricies duis. Curabitur pulvinar dignissim lorem. Cras pharetra conwallis est. Proin non turpis a turpis volutpat lacinia. Morbi eu elit eget elit lacinia dapibus. Phasellus eros tellus, rhoncus sagitta, vulputate eu, gravida ut, pede. Vivamus quis dolor. Vivamus ullamcorper sapien vel sapien. Curabitur rutrum. Suspendisse mauris purus, malesuada ac, dignibus et, feugiat in, enim. Aliquam id ipsum ut dolor pharetra pellentesque. Vestibulum porta, ante non fermentum consequat, eros nulla pellentesque ante, non vehicula tellus felis non libero. Cras eget mi. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vestibulum mattis ante vulputate libero. Ut varius, est sed faucibus viverra, purus orci euismod lorem, quis consectetur massa enim ac ante. Donec lacinia. In id pede. Duis mauris metus, gravida ac, fringilla ut, iaculis commodo, lectus. Vestibulum vitae libero. Maecenas lectus ipsum, euismod id, facilisis quis, faucibus at, tellus. Etiam sed felis. Proin malesuada massa at sapien. Maecenas wisi purus, condimentum in, suscipit in, commodo elementum, elit. Nullam blandit. Nulla scelerisque dolor id nulla. Duis justo. In hac habitasse platea dictumst. Etiam dignissim. Nullam vulputate turpis eu massa. Nam sem massa, placerat id, faucibus ut, gravida eget, metus. Sed porttitor dignissim sem. Curabitur nec libero non felis uacitor euismod. Mauris euismod pretium lacus. Vivamus rhoncus pulvinar diam. Curabitur cursus purus imperdiet massa consequat facilisis. Nulla porta condimentum nibh. Aenean enim. Nam mauris risus, consectetur id, consequat ut, posuere nec, sapien. Duis gravida. Phasellus tincidunt, nisi ut fermentum aliquet, veli nisi volutpat tellus, lobortis lacus lacus lorem sed ligula. Aliquam eu ligula. Duis lobortis interdum lectus. Suspendisse potenti. Aliquam erat volutpat. Nulla facilisi. Class



nulla. Vestibulum ac enim sit commodo. Morbi risus lacus, aliquam nec, lorem. Sed justo erat, tristique blandit, vel, duis. Fusce interdum porttitor. Fusce at nisl. Etiam auctor. Praesent vitae nisl at pulvinar iaculis nulla. Sed Curabitur id sem. Sed pretium libero. Praesent lacus. In hac Phasellus gravida pharetra commodo ut, congue quis, euismod, erat quis molestie wisi, non ullamcorper neque Sed commodo pellentesque iaculis est vitae justo.

© XMLMill XMLMill TestSuite

☞ In case the **absolute-position** attribute is fixed or absolute, the text-wrapping attribute is not taken into consideration (as this attribute only influences the elements in the normal flow).

6.2. Attribute: relative-position

Value: static | relative

Initial: static

Applies to: ml:table-and-caption | ml:table | ml:external-graphic | ml:textbox | ml:leader | ml:barcode

Inherited: no

Description:

The **relative-position** attribute defines that the element is part of the normal flow (of 'painting' elements in a region). This means they impact the layout/positioning of other elements displayed in the region.

Values have the following meaning:

static The element is placed in the normal flow of elements being placed in the region.

relative The element is placed in the normal flow of elements being placed in the region.

☞ *Currently there is no difference between the **static** and **relative** values.*

6.3. Attribute: wrap-option

Value: no-wrap | wrap

Initial: wrap

Applies to: ml:table-body | ml:table-row | ml:table-cell | ml:textbox

Inherited: yes (except for ml:textbox)

Description:

Specifies how line-wrapping (line-breaking) of the content of the formatting object is to be handled.

Values have the following meaning:

no-wrap No line-wrapping will be performed. In the case when lines are longer than the available width of the content-rectangle, the overflow will be treated in accordance with the "overflow" property specified on the element.

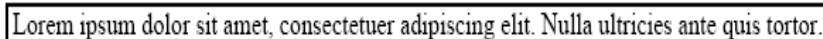
wrap Line-breaking will occur if the line overflows the available element's width. No special markers or other treatment will occur.

Example 1:

Following example will display a textbox with **wrap-option='no-wrap'**:

```
<ml:textbox border="1px solid black" width="80%">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Nulla ultricies ante quis tortor. Fusce eget lacus. Cum
  sociis natoque penatibus et magnis dis parturient montes,
  nascetur ridiculus mus.</ml:textbox>
```

The result is :



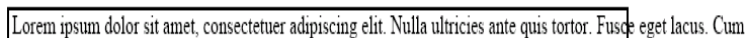
☞ *As the **wrap-option='no-wrap'** is defined, no wrapping is allowed (so only one line in printed).*

Example 2:

Following example will display a textbox with **wrap-option='no-wrap'** and **overflow="visible"**:

```
<ml:textbox border="1px solid black" width="80%" overflow="visible">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Nulla ultricies ante quis tortor. Fusce eget lacus. Cum
  sociis natoque penatibus et magnis dis parturient montes,
  nascetur ridiculus mus.</ml:textbox>
```

The result is :



☞ *As the **wrap-option='no-wrap'** is defined, no wrapping is allowed (so only one line in printed). As the **overflow** is visible, the text will go over the border of the textbox.*

☞ *Check the **Attribute_wrap-option-1.mill** and **Attribute_wrap-option-2.mill** examples in the **samples/mill** directory in the download for more examples.*

7. Bugfixes

Following bugfixes were solved:

- ◆ A background color for a `<ml:table-cell>` including an image was not painted. This has been solved.
- ◆ The height of a `<ml:table-cell>` with `width="0%"` inside a `<ml:table>` with `table-layout="complex"` was expanded to the end of the page, making the cell too large. This has been solved.
- ◆ Using a `number-rows-spanned` attribute in collaboration with a `width="0%"` attribute inside a `<ml:table-cell>` could sometimes, after many minutes, end in an `com.xmlmill.exception.XMLMillFatalError` error. This has been solved.
- ◆ Eliminate the `java.lang.ArrayIndexOutOfBoundsException: 256` that could appear when using external `Type1` fonts.
- ◆ Images (`ml:external-graphic`) will no longer be resized at the end of a page (previously they always fitted on the page). A new page is added and the image is printed on the next page (on the condition the `content-height` has been defined).
- ◆ The `overflow='visible'` attribute of a region element (`ml:region-body`, `ml:region-before`, `ml:region-after`, `ml:region-start` and `ml:region-end`) was not applied on a textbox in case a textbox was absolutely positioned in the region.
- ◆ Sometimes an additional `java.lang.ArrayIndexOutOfBoundsException` occurred following an occurrence of an `XMLMillException` error. This has been solved.
- ◆ The `border-style` and `border-width` attributes of a `<ml:table-cell>` did not change the appearance of the cell. This has been solved.
- ◆ Several bugs were detected when using external `Type1` fonts, like the presence of 'garbage' characters and wrong character spacing.

XML Mill