

---

XMLMill

---

Release Notes

**XMLMill v 1.50**

**1.0**

xmlmill

© 2000-2003 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. (First printing, November 2003)

# Table of Contents

<b>1. Preface.</b>	<b>6</b>
<b>2. Upgrading from previous versions.</b>	<b>7</b>
2.1. Upgrading from 1.40.	7
2.2. Upgrading from earlier versions.	7
<b>3. XMLMill Logging.</b>	<b>8</b>
3.1. LogLevels.	8
3.2. Default Logging behavior.	9
<b>4. XMLMill Namespace.</b>	<b>11</b>
4.1. Scenario 1: XML contains data, XSL contains transformation instructions (XMLMill namespace).	11
4.2. Scenario 2: XML contains data, XSL contains transformation instructions (default namespace).	13
4.3. Scenario 3: mill document containing data + transformation instructions (XMLMill namespace).	13
4.4. Scenario 4: mill document containing data + transformation instructions (default namespace).	14
<b>5. XMLMill Barcode.</b>	<b>16</b>
5.1. Barcode : type39.	16
5.1.1. Attribute: type.	16
5.1.2. Attribute: value.	17
5.1.3. Attribute: add-check-digit.	17
5.1.4. Attribute: add-check-digit-to-text.	17
5.1.5. Attribute: show-start-stop-character.	17
5.1.6. Attribute: rotation.	18
5.1.7. Attribute: narrow-bar-width.	18
5.1.8. Attribute: wide-bar-multiplier.	19
5.1.9. Attribute: bar-height.	19
5.1.10. Attribute: bar-color.	19
5.2. Barcode : type39 Extended.	19
<b>6. XMLMill Type1 Fonts.</b>	<b>20</b>
6.1. Overview.	20
6.1.1. Encoding: Unicode.	20
6.2. Font Embedding and Subsetting.	20
6.2.1. Font Embedding.	20
6.2.1.1. Standard fonts.	20
6.2.1.2. External Type1 fonts.	21
6.2.2. Font kerning.	21
6.2.3. Font Subsetting.	22
6.2.4. Ligatures.	22
6.2.5. composite characters.	22
<b>7. XMLMill Configuration.</b>	<b>23</b>
7.1. Introduction.	23
7.2. Location of config files.	23
7.2.1. Built-in config.xml file.	23
7.2.2. External config.xml file.	23

7.3. Content of a of config.xml file. ....	23
7.3.1. tag: <pagetemplate>. ....	23
7.3.2. tag: <font>. ....	24
7.3.2.1. tag: <standard>. ....	24
7.3.2.2. tag: <external>. ....	25
7.3.3. tag: <glyphlist>. ....	25
<b>8. Java API. ....</b>	<b>26</b>
8.1. Class: PDXLogFile. ....	26
8.2. Class: PDXLogger. ....	26
8.2.1. Default Logging behavior. ....	26
8.2.2. public static void setLevel(org.apache.log4j.Level ). ....	27
8.2.3. public static void setLogger(org.apache.log4j.Logger). ....	27
8.2.4. public static void resetLogger(). ....	27
8.3. Class: PDXTransform. ....	27
8.3.1. setFeature(String, boolean). ....	27
8.4. Class: PDXTransformErrorHandler. ....	27
8.4.1. Fields. ....	27
8.4.2. void PDXTransformErrorHandler(). ....	28
8.4.3. boolean isPrintWarning(). ....	28
8.4.4. boolean isPrintError(). ....	28
8.4.5. boolean isPrintStackTrace(). ....	28
8.5. Class: Configurator. ....	28
8.5.1. How to use the Configurator. ....	28
8.5.1.1. Do not call the Configurator. ....	28
8.5.1.2. Call the Configurator with no parameter.. ....	29
8.5.1.3. Call the Configurator passing a parameter. ....	29
8.5.2. Method: static synchronized void configure(). ....	29
8.5.3. Method: static synchronized void configure(String s). ....	29
8.5.4. Method: static synchronized void configure(URL url). ....	29
8.5.5. Method: static synchronized void configure(File f). ....	29
8.5.6. Method: static void setErrorHandler(PDXTransformErrorHandler handler). . .	29
<b>9. XMLMill batch. ....</b>	<b>30</b>
9.1. Arguments. ....	30
9.1.1. Suppressed. ....	30
9.1.2. New. ....	30
<b>10. XMLMill Interactive. ....</b>	<b>31</b>
10.1. Graphical User Interface. ....	31
10.1.1. Log. ....	31
10.1.2. Configuration. ....	32
<b>11. Disclaimer. ....</b>	<b>33</b>

xmlmill

## 1. Preface

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

This release notes document describes all new (or enhanced) functionalities of version 1.50 of XMLMill.

The main changes are:

- ◆ The logging model is now based on the industry-standard **Log4j** package.
- ◆ All XMLMill tags are in a separate namespace: **ml**, facilitating largely the construction of xsl stylesheets using your favorite xml-editor.
- ◆ Different types of barcode can be added in your document, **without** the use of barcode-fonts.
- ◆ **Type1** fonts can be added to XMLMill when you want to generate documents with your own fonts.
- ◆ An external configuration file can be used to define some standard values (like margins, media, default font) but also the representation of **type1** fonts.

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

- ◆ **apidoc/** -- The JavaDoc concerning the **PDX** api.
- ◆ **userguide.pdf** -- XMLMill user's guide.
- ◆ **dtdguide.pdf** -- An explanation of XMLMill's tag and their attributes.

*If you are new to XMLMill you can skip this document and read the **userguide.pdf**. You will find all information necessary to get started.*

If you have questions, please do not hesitate to send a mail to [support@xmlmill.com](mailto:support@xmlmill.com).

☞ *This document is completely generated with XMLMill 1.50 using **rnotes150.xml** and **rnotes.xsl**. These files can be found in the directory **samples/docs/rnotes** in the download.*

## 2. Upgrading from previous versions

### 2.1. Upgrading from 1.40

As of this version the logging is based on the industry-standard **Log4j** package. In order to successfully run XMLMill you need to do following:

1. Download the latest **log4j** distribution at <http://jakarta.apache.org/log4j/docs/index.html> (in case you do not use **log4j** yet).
2. Copy the **log4j-1.2.8.jar** file (or similar) to the **lib/** directory (created when you extracted the XMLMill distribution).
3. Rename the **log4j-1.2.8.jar** file (or similar) to **log4j.jar**

☞ *In case you already use a previous version of **log4j**, you can use that version.*

☞ *In case the **log4j** jar file has a different name than **log4j.jar**, please modify the **CLASSPATH** in the scripts in the **etc/** directory accordingly.*

☞ *In case you start XMLMill interactive in Windows by double-clicking the **xmlmill.jar** file and the **log4j** jar file has a different name than **log4j**, please modify the **manifest.mf** file accordingly.*

No other changes are required.

### 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 files.
2. Read all release notes [online](#) from your current version to the latest version in correct order (from oldest release notes to newest).
3. Modify your .xml / .mill / .xsl files accordingly to the release notes.

## 3. XMLMill Logging

As of this version the logging is based on the industry-standard **Log4j** package. The **Log4j** package allows the developer to control which log statements are logged with arbitrary granularity. It is fully configurable at runtime using external configuration files. This implies that:

- ◆ XMLMill's logging behavior has been changed.
- ◆ It is possible to use your own **org.apache.log4j.Logger** instance.

### 3.1. LogLevels

As of this version all messages have a **loglevel**, corresponding to the 5 normal levels available in the **org.apache.log4j.Level** class and two special levels of logging:

1. **org.apache.log4j.Level.DEBUG**
2. **org.apache.log4j.Level.INFO**
3. **org.apache.log4j.Level.WARN**
4. **org.apache.log4j.Level.ERROR**
5. **org.apache.log4j.Level.FATAL**
6. **org.apache.log4j.Level.ALL**
7. **org.apache.log4j.Level.OFF**

☞ *For more information regarding these **logLevels**, please refer to the **Log4j** documentation: <http://jakarta.apache.org/log4j/docs/index.html>.*

Following messages correspond to following loglevels:

#### **org.apache.log4j.Level.DEBUG**

- ◆ Comments between **<comment>... </comment>** tags.
- ◆ **StackTraces** generated by **javax.xml.transform.TransformerException** or **org.xml.sax.SAXException** exceptions.

#### **org.apache.log4j.Level.WARN**

- ◆ **com.xmlmill.XMLMillWarning** messages.
- ◆ **javax.xml.transform.TransformerException (warning)** messages.
- ◆ **org.xml.sax.SAXException (warning)** messages.

#### **org.apache.log4j.Level.ERROR**

- ◆ **com.xmlmill.XMLMillError** messages.

- ◆ **javax.xml.transform.TransformerException (error)** messages.
- ◆ **org.xml.sax.SAXException (error)** messages.

### org.apache.log4j.Level.FATALERROR

- ◆ **com.xmlmill.XMLMillFatalError** messages.
- ◆ **javax.xml.transform.TransformerException (fatalerror)** messages.
- ◆ **org.xml.sax.SAXException (fatalerror)** messages.

### org.apache.log4j.Level.INFO

- ◆ All other messages.

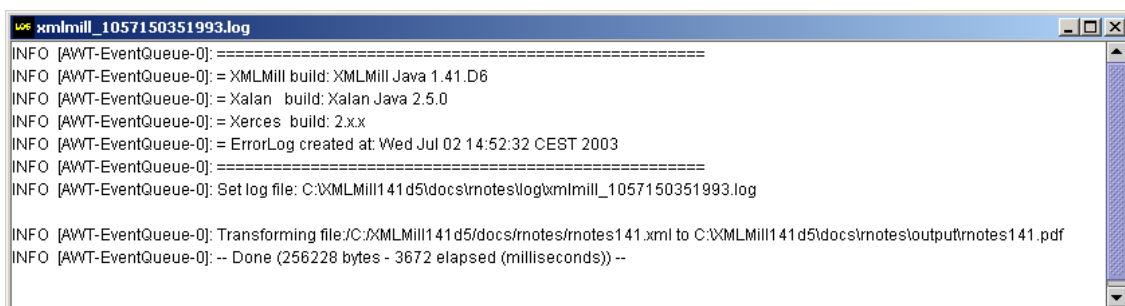
☞ *The levels are hierarchical defined. Following rule applies: **ALL** < **DEBUG** < **INFO** < **WARN** < **ERROR** < **FATAL** < **OFF**. Only output messages that are of a level greater than or equal to the level set will be logged. If the level is set at **INFO**, alle messages will be logged accept the messages with level **DEBUG**.*

## 3.2. Default Logging behavior

If no logfile or logger instance is defined, the default logging will apply:

- ◆ All logmessages are send to the standard error stream (**System.err**).
- ◆ All logmessages with a level of at least **org.apache.log4j.Level.INFO** will logged.
- ◆ A default layout will be used.

A **.log** file could look like this:



```

xmlmill_1057150351993.log
INFO [AWT-EventQueue-0]: =====
INFO [AWT-EventQueue-0]: = XMLMill build: XMLMill Java 1.41.D6
INFO [AWT-EventQueue-0]: = Xalan build: Xalan Java 2.5.0
INFO [AWT-EventQueue-0]: = Xerces build: 2.x.x
INFO [AWT-EventQueue-0]: = ErrorLog created at: Wed Jul 02 14:52:32 CEST 2003
INFO [AWT-EventQueue-0]: =====
INFO [AWT-EventQueue-0]: Set log file: C:\XMLMill\141d5\docs\notes\log\xmlmill_1057150351993.log

INFO [AWT-EventQueue-0]: Transforming file:J:\XMLMill\141d5\docs\notes\notes141.xml to C:\XMLMill\141d5\docs\notes\output\notes141.pdf
INFO [AWT-EventQueue-0]: -- Done (256228 bytes - 3672 elapsed (milliseconds)) --
    
```

The default logging will contain following information:

1. The log level.
2. The thread-name.
3. The log message.

☞ *To know how to overrule this default logging behavior please visit the **PDXLogger** class in the [Java API](#) chapter.*

xmlmill

## 4. XMLMill Namespace

As of this version XMLMill has its own namespace: **ml**. This largely facilitates developing XSL stylesheets using the XMLMill tags and appropriate attributes.

To be compatible with previous versions you are still allowed to omit the namespace, this means that the XMLMill tags are in the default namespace.

To summarize, following different scenario's are possible:

	<b>XML Document (containing data)</b>	<b>XSL Document with default namespace</b>	<b>XSL Document with XMLMill namespace defined</b>	<b>MILL document with default namespace</b>	<b>MILL document with XMLMill namespace defined</b>
<b>Scenario 1:</b> XML contains data, XSL contains transformation instructions	X		X		
<b>Scenario 2:</b> XML contains data, XSL contains transformation instruction	X	X			
<b>Scenario 3:</b> MILL document with default namespace containing data + formatting instructions				X	
<b>Scenario 4:</b> MILL document with XMLMill namespace containing data + formatting instructions					X

### 4.1. Scenario 1: XML contains data, XSL contains transformation instructions (XMLMill namespace)

Namespaces are implemented by attaching a prefix to each element and attribute. Each prefix is mapped to a URI by an **xmlns:prefix** attribute. Default URIs can also be provided for elements that don't have a prefix by xmlns attributes. Elements and attributes that are attached to the same URI are in the same namespace.

In order to use the namespace it needs to be defined in the xsl stylesheet. Following information needs to be added to the xsl stylesheet:

```
<?xml version="1.0" encoding="UTF-16" ?>
<xsl:stylesheet version="1.0"
  xmlns:ml="http://www.xmlmill.com/XSL/Format"
```

```

❶ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
❷ xsi:schemaLocation="http://www.xmlmill.com/XSL/Format xmlmill.xsd"
❸
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:output method="xml" indent="yes" encoding="UTF-16" />
<xsl:template match="/">
  <ml:documents linefeed-treatment="ignore">
    <ml:document file ="output\phonelist.pdf">
      <ml:info>
        ...

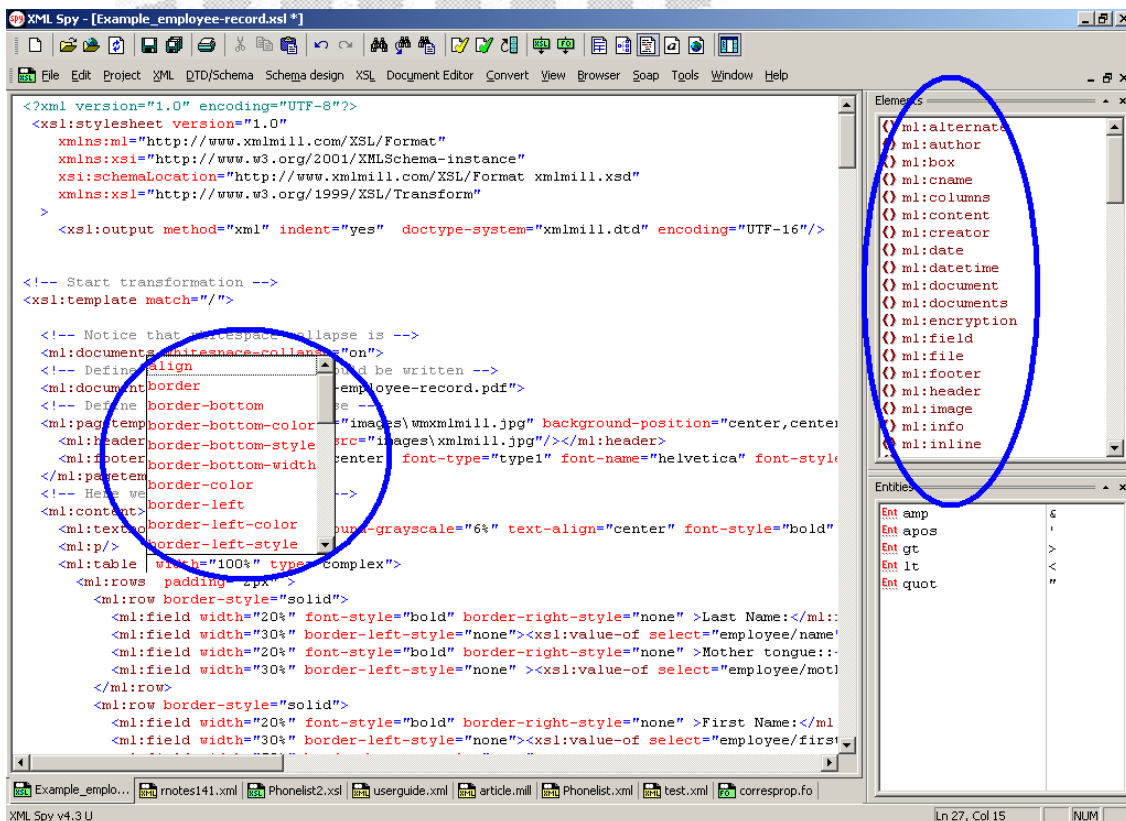
```

- ❶ Define the **ml** namespace by referring to the correct namespace name. In this case the namespace name is a URI mapping to: **http://www.xmlmill.com/XSL/Format**.
- ❷ Define that an instance of a XMLSchema will be defined.
- ❸ Define where the instance of the XMLSchema is located. The XMLSchema is referenced using the namespace name (**http://www.xmlmill.com/XSL/Format**). The second parameter (**xmlmill.xsd**) defines the actual location and name of .xsd schema.

When the xml document is transformed with the xsl stylesheet, an internal **.mill** document is generated, used to generate the .pdf document.

*The docs/xsd/ directory contains a default xsl stylesheet and the **xmlmill.xsd** and **xslt.xsd** file. Please keep these two files in the same directory as the **xslt.xsd** file is referenced in the **xmlmill.xsd** file.*

Using a namespace facilitates developing xsl stylesheet, as your xml editor will control which attributes are valid for a tag:



The example above shows how the attributes of an XMLMill tag appears when a `<table>` tag is entered.

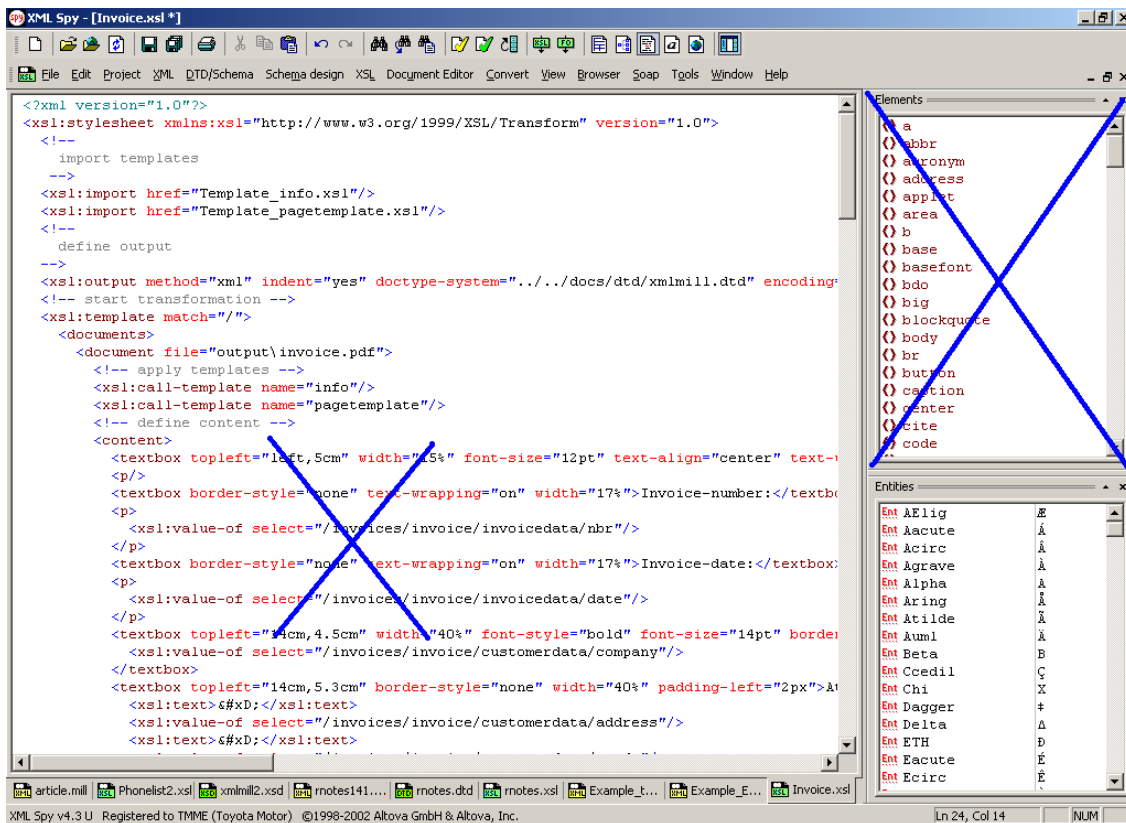
## 4.2. Scenario 2: XML contains data, XSL contains transformation instructions (default namespace)

This was the standard scenario up to version 1.40. The XMLMill tags are in the default namespace:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes" encoding="UTF-8" />
<xsl:template match="/">
<documents>
<document file="output\example_table_complex.pdf">
<content>
...
```

*Note that the **ml:** namespace name is absent in the tag names.*

The disadvantage is that your xml-editor can't give any support concerning which tags and attributes can be used, as no XMLSchema is known:



Above example shows that no support is available when the stylesheet is developed.

*As of this version always use the namespace as explained in scenario 1.*

## 4.3. Scenario 3: mill document containing data + transformation instructions (XMLMill namespace)

In case you want to generate directly a **mill** document without using a .xsl sheet, start your **.mill** document with following:

```
<?xml version="1.0" encoding="UTF-16"?>
<ml:documents
  xmlns:ml="http://www.xmlmill.com/XSL/Format" ❶
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ❷
  xsi:schemaLocation="http://www.xmlmill.com/XSL/Format xmlmill.xsd" > ❸
<ml:document file="output\article.pdf">
  <ml:info>
    ...
```

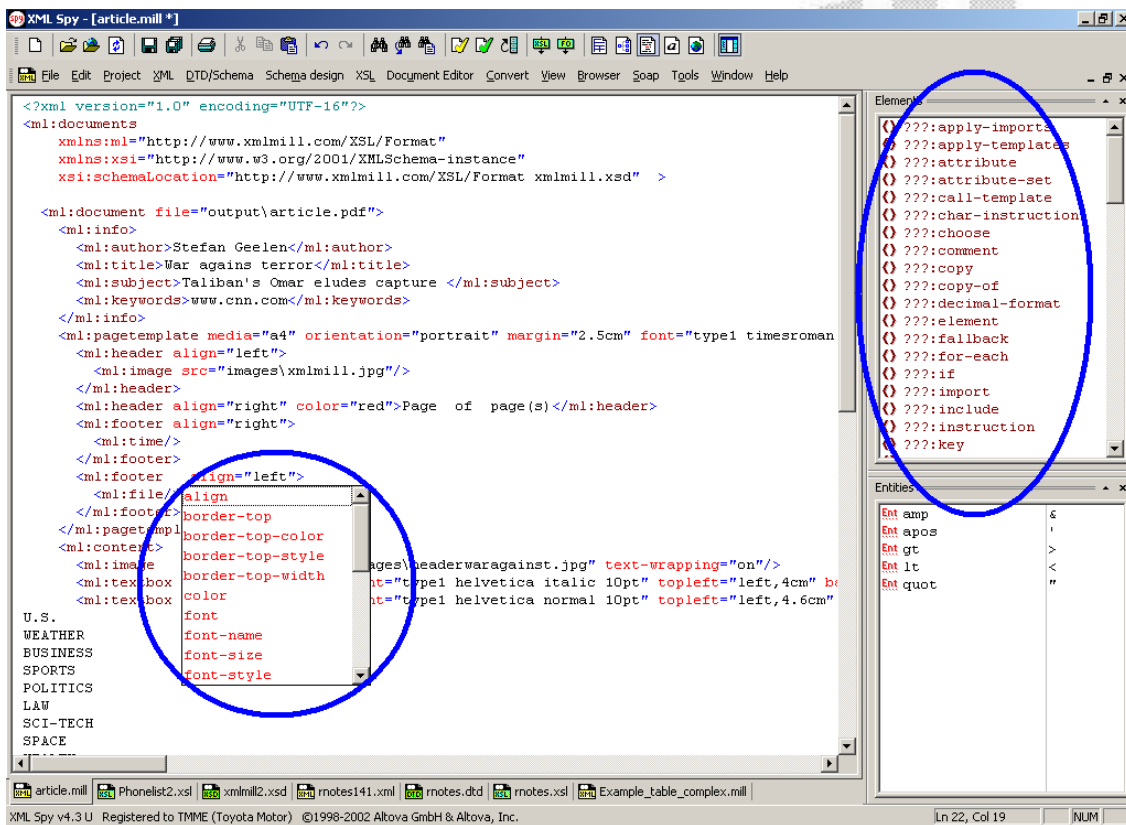
❶ Define the **ml** namespace by referring to the correct namespace name. In this case the namespace name is a URI mapping to: **http://www.xmlmill.com/XSL/Format**.

❷ Define that an instance of a XMLSchema will be defined.

❸ Define where the instance of the XMLSchema is located. The XMLSchema is referenced using the namespace name (**http://www.xmlmill.com/XSL/Format**). The second parameter (**xmlmill.xsd**) defines the actual location and name of .xsd schema.

☞ Note that the **xsl** namespace declaration (**xmlns:xsl="http://www.w3.org/1999/XSL/Transform"**) is absent.

Using a namespace facilitates the development of an internal **.mill** document as your xml editor will control which attributes are valid for a tag:



☞ Note that the **xsl** prefix is not selectable (replaced by **???**) as in a **mill** document no transformation instructions are allowed.

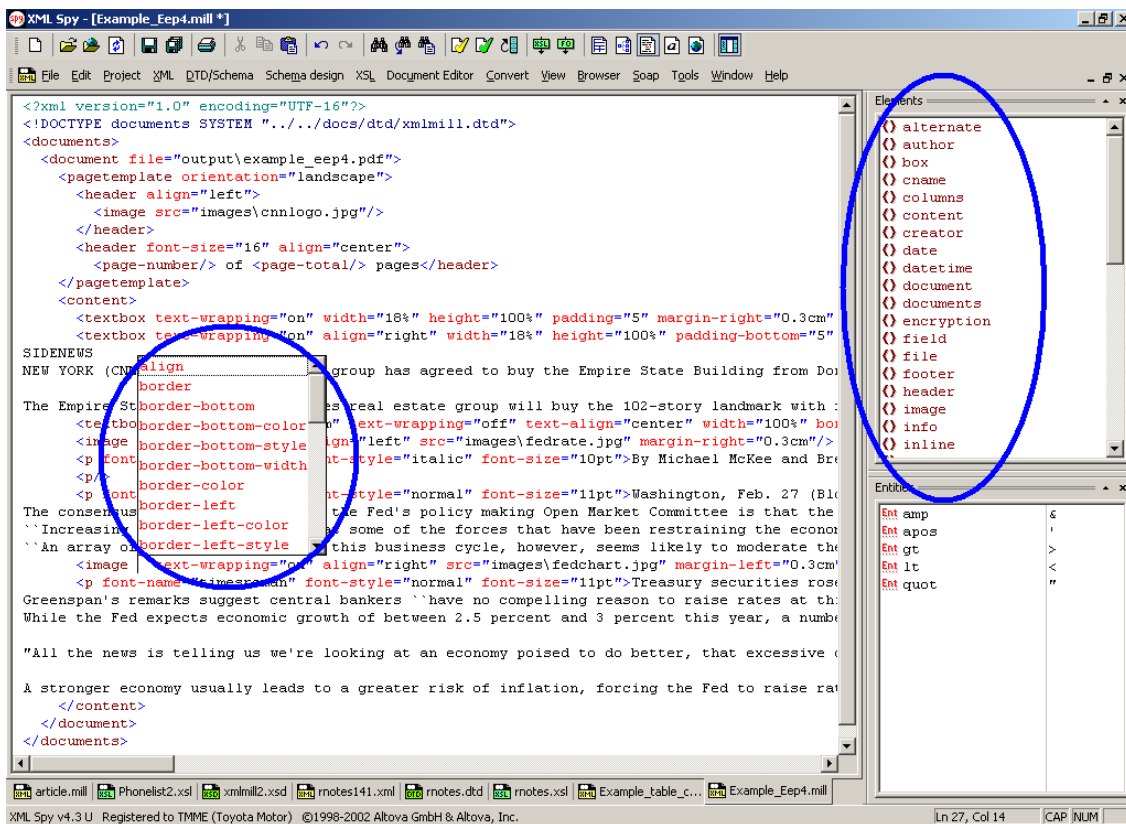
#### 4.4. Scenario 4: mill document containing data + transformation instructions (default namespace)

This was the standard scenario up to version 1.40. Use the `xmlmill.dtd` document to define XMLMill tags and attributes.

```
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE documents SYSTEM "../..//docs/dtd/xmlmill.dtd">
<documents>
  <document file="output\example_eep4.pdf">
    <pagetemplate orientation="landscape">
      <header align="left">
        ...
```

❶ Define the **dtd** document referring to **xmlmill.dtd** file

Using the `xmlmill.dtd` document facilitates developing xsl stylesheet, as your xml editor will control which attributes are valid for a tag:



☞ *Note that no separate namespace is used.*

☞ *As of this version always use the namespace as explained in scenario 3 (if you want to define your own .*

## 5. XMLMill Barcode

As of this version you can directly generate barcodes in your PDF document without the use of barcode fonts. This allows you to generate barcodes even on a Linux/Unix server without any graphical (X-server) server installed.

Currently following barcode types are available:

- ◆ type 39
- ◆ type 39 - extended

☞ *In the next version more types will be added. Please send a mail to [support@xmlmill.com](mailto:support@xmlmill.com) to request additional barcodes if needed.*

### 5.1. Barcode : type39

This is the implementation of the 3 of 9 standard (also commonly called LOGMARS, 'Code 3 of 9' or the '3 of 9 Code' or 'Code39'), a widely used barcode standard that includes capital letters, numbers, and several symbols. Following characters are allowed:

**0123456789ABCDEFGHIJKLMNQRSTUUVWXYZ-. \$/+%\*.**

To create a valid 3 of 9 barcode you have to begin and end it with a special character. Scanners look for this character to know where to start and stop reading the barcode. It is represented by the '\*' character. XMLMill will automatically add the '\*' to the text.

Following specific barcode attributes are available:

#### 5.1.1. Attribute: **type**

Defines which type of barcode should be generated.

#### Values

- ◆ **code39**
- ◆ **code39e**
- ◆ **upca** (*not yet supported*)
- ◆ **upce** (*not yet supported*)
- ◆ **supp2** (*not yet supported*)
- ◆ **supp5** (*not yet supported*)
- ◆ **postnet** (*not yet supported*)
- ◆ **planet** (*not yet supported*)
- ◆ **code128** (*not yet supported*)
- ◆ **code128\_ucc** (*not yet supported*)

- ◆ **code128\_raw** (*not yet supported*)
- ◆ **codabar** (*not yet supported*)
- ◆ **ean8** (*not yet supported*)

### Example

```
<ml:barcode type="code39" value="XMLMill - 123456">
```

#### 5.1.2. Attribute: **value**

Defines the text to encode.

#### Value

**xx99** A value (numeric and alphanumeric) indicating the text to encode.

```
<ml:barcode type="code39e" value="XMLMill - 123456">
```

#### 5.1.3. Attribute: **add-check-digit**

Defines if a checkdigit should be added to the value in order to calculate the barcodes. The checkdigit is calculated as the modulus 43 of the value.

#### Values

- ◆ **on**
- ◆ **off** (default)

### Example

```
<ml:barcode type="code39" value="XMLMill - 123456" add-check-digit="on">
```

#### 5.1.4. Attribute: **add-check-digit-to-text**

Defines if a checkdigit should be added to the text that is printed below the barcode (if any).

#### Values

- ◆ **on**
- ◆ **off** (default)

### Example

```
<ml:barcode type="code39" value="XMLMill - 123456" add-check-digit-to-text="on" >
```

#### 5.1.5. Attribute: **show-start-stop-character**

Defines if the start-stop characters (the '\*') should be added to the text that is printed below the barcode (if any).

#### Values

- ◆ **on**
- ◆ **off** (default)

#### Example

```
<ml:barcode type="code39" value="XMLMill - 123456" add-check-digit-to-text="on" >
```

#### 5.1.6. Attribute: **rotation**

Defines how the barcode should be painted on the page.

#### Values

- ◆ **0** -- The barcode is not rotated on the page (default).
- ◆ **90** -- The barcode is 90 degrees clock-counter wise rotated on the page.
- ◆ **180** -- The barcode is 180 degrees clock-counter wise rotated on the page (the barcode is printed upside down).
- ◆ **270** -- The barcode is 270 degrees clock-counter wise rotated on the page.

#### Example

```
<ml:barcode type="code39" value="XMLMill - 123456" rotation="90" >
```

#### 5.1.7. Attribute: **narrow-bar-width**

Defines the size of narrow bar (default value: 1px). The narrow bar width is also called the x-dimension of a barcode.

#### Values

- 99** Measurement in pixels (px omitted).
- 99px** Measurement in pixels.
- 99in** Measurement in inches.
- 99cm** Measurement in centimeters.
- 99mm** Measurement in millimeters.

#### Example

```
<ml:barcode type="code39" value="XMLMill - 123456" narrow-bar-width="1cm" >
```

☞ *Before attempting to scan barcodes that are printed with a small narrow-bar-width value, check*

*your barcode scanner manual to make sure your scanner can read it. Also, make sure your printer can accurately reproduce barcodes at these sizes.*

#### 5.1.8. Attribute: **wide-bar-multiplier**

Defines the size of the wide-bar (defined as a multiple of the narrow bar width).

##### Values

- ◆ **99.99** -- A number defining the multiplier (default value: 2 ).

```
<ml:barcode type="code39" value="XMLMill - 123456" wide-bar-multiplier="1.5" >
```

#### 5.1.9. Attribute: **bar-height**

Defines the height of the bars. The default value is the font-size of the text multiplied by 3.

##### Values

- 99** Measurement in pixels (px omitted).
- 99px** Measurement in pixels.
- 99in** Measurement in inches.
- 99cm** Measurement in centimeters.
- 99mm** Measurement in millimeters.

```
<ml:barcode type="code39" value="XMLMill - 123456" bar-height="2cm" >
```

#### 5.1.10. Attribute: **bar-color**

Defines the color of the bars (default: black).

##### Values

- ◆ **#999999** -- A hexadecimal number representing the color.
- ◆ **named color** -- A [named color](#).
- ◆ **rgb(99,99,99)** -- A rgb color (ref, green, blue).

```
<ml:barcode type="code39" value="XMLMill - 123456" bar-color="red" >
```

## 5.2. Barcode : **type39 Extended**

This is the implementation of the extended 3 of 9 standard (sometimes called "code 39"). Extended Code 39 encodes the full 128 character ASCII character set.

```
<ml:barcode type="code39e" value="XMLMill - test - 123456" >
```

For more information please visit the [code39 section](#).

## 6. XMLMill Type1 Fonts

As of this version XMLMill supports **external** Type1 fonts, which can be embedded and/or subsetted in the generated PDF document.

This chapter explains the functionality XMLMill offers regarding **standard** Type1 fonts and **external** Type1 fonts. The [configuration](#) chapter explains the various configuration options regarding these functionalities.

☞ *TrueType fonts will be added in a next version of XMLMill.*

### 6.1. Overview

Font handling is one of the most complex aspects when generating a PDF document. Due to the intelligent behavior of XMLMill the mapping between the initial encoding used in a xml/xsl document and the encoding used in a PDF document will be transparent for the user.

#### 6.1.1. Encoding: Unicode

When a XML document is converted to a PDF document, the xml/xsl encoding is converted to UTF-16 (ISO-10646 codepage). In a next step these encodings will be mapped to the encoding defined in a font-metrics file. As a result the user does not need to know which glyph is represented by which character in the font-metrics file. Moreover, *switching from one font to another should not be any problem* (as long as the requested character is available in the font-file).

☞ *If a character does not exist in the font-file, a '?' (question mark) is shown. If the question mark is not defined in the font, the font's .notdef character will be displayed.*

### 6.2. Font Embedding and Subsetting

#### 6.2.1. Font Embedding

##### 6.2.1.1. Standard fonts

PDF viewers support a core set of 14 **standard** fonts which are guaranteed to be available in all PDF viewer applications. These fonts include four faces of each of three Latin text typefaces (Courier, Helvetica and Times) and two symbolic fonts (Symbol and ITC ZapfDingbats).

These fonts have always been supported since version 1.0 of XMLMill. They are represented using a correct **font-type**, **font-name** and **font-style** attribute (where applicable), as indicated in following table:

PostScript name	font-type	font-name	font-style
Courier	type1	courier	normal
Courier-Bold	type1	courier	bold
Courier-Oblique,	type1	courier	italic
Courier-BoldOblique,	type1	courier	bolditalic

PostScript name	font-type	font-name	font-style
Helvetica	type1	helvetica	normal
Helvetica-Bold	type1	helvetica	bold
Helvetica-Oblique	type1	helvetica	italic
Helvetica-BoldOblique	type1	helvetica	bolditalic
Times-Roman	type1	timesroman	normal
Times-Bold	type1	timesroman	bold
Times-Italic	type1	timesroman	italic
Times-BoldItalic,	type1	timesroman	bolditalic
Symbol	type1	symbol	normal
ZapfDingbats	type1	zapfdingbats	normal

### Example:

```
<p font-type="type1" font-name="helvetica" font-style="bold" font-size="8pt">Some text</p>
```

As these fonts are guaranteed available by each viewer no font-metrics or font-program information is added to the PDF document. As a result, PDF documents generated with **onlystandard** fonts will have the smallest size possible.

- ☞ *Note that the fontnames are all in lowercase when defined with the **font-name** attribute (for compatibility with previous versions of XMLMill).*
- ☞ *You are allowed to change the values off the attributes by changing these values in the [configuration file](#).*

#### 6.2.1.2. External **Type1** fonts

As of this version you can use **external** Type1 fonts to make-up PDF documents. XMLMill is capable of embedding font-metrics and font-programs into the generated PDF output.

If an **external** Type1 font is not embedded, Acrobat will take it from the target system if available, or construct a substitute font according to the font descriptor in the PDF.

To use an external **Type1** font you need:

- ◆ the corresponding font-metrics file (**.afm** file).
- ◆ the corresponding font-program file (**.pfb** or **.pfa** file).

- ☞ *Support for the **.pfm** file-type will be added in a next version of XMLMill.*

#### 6.2.2. Font kerning

Font kerning is the process of determining the spacing adjustments between characters depending on context.

Font kerning is optional and may or may not be present for a given font. Kerning data is supplied in two forms: **track-kerning** and **pair-wise-kerning**. Track kerning is applied to all characters uniformly, whereas pair-wise kerning is applied to specific character pairs. Track kerning and pair-wise kerning can be used independently or together (that is, it is possible to apply track kerning to a line of text and then to apply pair-wise kerning).

☞ *XMLMill supports currently only **pair-wise-kerning**. **Track-kerning** will be added in a next version of XMLMill.*

### 6.2.3. Font Subsetting

In order to decrease the size of the PDF output, XMLMill can embed only those characters from a font which are actually used in the document. This process is called font subsetting.

### 6.2.4. Ligatures

Ligatures are currently not supported.

### 6.2.5. composite characters

Composite characters are currently not supported.

☞ *External **Type1** fonts can be configured by modifying the `<external>` tag in the [configuration file](#).*

xmlmill

## 7. XMLMill Configuration

### 7.1. Introduction

As of this version XMLMill has its own configuration file. This file is mainly used for:

- ◆ Overriding default values for **<pagetemplate>** settings (margins, default font, ...).
- ◆ Defining the font specifications of **external** Type1 fonts.
- ◆ Defining the location of the Glyphlist files.

### 7.2. Location of config files

#### 7.2.1. Built-in config.xml file

The **xmlmill.jar** file contains a default configuration file that is used if no external configuration file is defined. This allows you to embed the **xmlmill.jar** file in your application (if you are a reseller for example) with the correct configuration parameters, without the need to pass an external configuration file. In the jar file you will find the file in the **com\xmlmill\conf\** directory, called **config.xml**.

#### 7.2.2. External config.xml file

It is probably easier to use an external configuration file. A **config.xml** file is available in the **conf/** directory contained the download.

☞ *It is allowed to rename this file if needed. The only limitations are that it must be a well-formed and valid xml file.*

### 7.3. Content of a of config.xml file

A **config.dtd** is available in the **conf/** directory that defines the allowed tags/attributes in a **config.xml** file.

The configuration file has following main tags:

- ◆ **<pagetemplate>** -- defines the default pagetemplate setting if these are not define at document level.
- ◆ **<font>** -- defines the characteristics of the **standard** fonts and the **external** fonts.
- ◆ **<glyphlist>** -- defines the location of the **default** glyphlist file and the **optional** glyphlist file.

The following sections explain these tag in more details.

#### 7.3.1. tag: **<pagetemplate>**

The **<pagetemplate>** tag defines the values of the attributes that are used when a document is generated without a **<pagetemplate>** tag or with not all attributes defined.

The default configuration file following attribute/value pair regarding this tag:

```
<pagetemplate media="a4" orientation="portrait"
  font-type="type1" font-name="helvetica" font-size="10pt"
  font-style="normal"
  margin-top="2.5cm" margin-right="2.5cm" margin-bottom="2.5cm"
  margin-left="2.5cm"/>
```

### 7.3.2. tag: <font>

The fonts section describes:

- ◆ The attributes of the **standard** fonts.
- ◆ The attributes of the **external** fonts.

#### 7.3.2.1. tag: <standard>

This tag contains a number of <font> tags describing the attributes of each internal font.

```
<font font-type="type1" track-kerning="on" pair-wise-kerning="on"
  font-metrics-file="jar:com/xmlmill/resources/afm/hv____.afm"
  font-program-file="jar:com/xmlmill/resources/afm/hv____.afm">
  <font-info font-name="helvetica" font-style="normal"/>
</font>
```

The attributes are:

- font-type** Describing the type of the font (should always be **type1**).
- track-kerning** Defines if track-kerning should be 'on' or 'off'.
- pair-wise-kerning** Defines if pair-wise-kerning should be 'on' or 'off'.
- font-metrics-file** Describes the location of the fontmetrics file (for type1 fonts, this is a **.afm** file).
- font-program-file** Describes the location of the fontprogram file (for type1 fonts, this is a **.pfb** or **.pfa** file).

☞ You should always reference the **font-metrics-file** and **font-program-file** using a valid URL.

☞ If the **font-metrics-file** value and **font-program-file** start with the **jar** protocol, it is assumed that the path references a path in an existing **.jar** file. By default all **.afm** files of the built-in internal fonts are located in the **xmlmill.jar** file at: **com/xmlmill/resources**.

☞ Be aware that defining the **font-metrics-file** value and **font-program-file** filenames are case-sensitive when referred to a **jar**: path.

Each <font> tag needs to contain at least one <font-info> tag describing the attributes/value pairs needed in a .mill file to select the font to use. These attributes are:

- ◆ **font-name** -- The name of the font.
- ◆ **font-style** -- The style of the font.

☞ *The **font-name** and **font-style** should together uniquely define a font (per font-type). If not, XMLMill will not be able uniquely define the required font during transformation.*

### 7.3.2.2. tag: `<external>`

This tag contains a number of `<font>` tags describing the attributes of each external font.

```
<font font-type="type1" embed="on" subset="on"
      track-kerning="on" pair-wise-kerning="on"
      font-metrics-file="file:/C:/XMLMill150/fonts/caligula/CALIG____.AFM"
      font-program-file="file:/C:/XMLMill150/fonts/caligula/CALIG____.PFB">
  <font-info font-name="Calig" font-style="normal"/>
</font>
```

Following additional attributes are:

- ◆ **embed** -- If the font-metrics and font-outlines should be embedded in the PDF document.
- ◆ **subset** -- If the font-outlines should be subsetted in the PDF document.

In order to configure XMLMill with the contents of a configuration file please visit the [Class: Configurator](#) section.

### 7.3.3. tag: `<glyphlist>`

The `glyphlist` tag contains the unicode-to-glyphname list. This list determines how each unicode character is mapped to a "glyph" (a graphical shape representing a character). This allows XMLMill to replace the unicode characters in a xml/xsl document by the correct characters (bytes) that are included in the PDF document.

```
<glyphlist default="jar:com/xmlmill/resources/glyphlist/glyphlist_in_mem.txt"
           optional="jar:com/xmlmill/resources/glyphlist/glyphlist.txt"/>
```

This tag has two attributes:

- default** Describes the location of the file containing the list of unicode-to-glyphname comparisons that are read into memory. The default list (**com/xmlmill/resources/glyphlist/glyphlist\_in\_mem.txt**) contains all the WinAnsi characters.
- optional** Describes the location of the file containing the **complete** list of unicode-to-glyphname comparisons. If, during the transformation process, a character is needed that is not defined in the 'default' glyphlist, the character will be read from this list.

☞ *You should always reference the **default** and **optional** files using a valid URL.*

☞ *You are free to change the content of both files.*

☞ *The default configuration will use the glyphlists defined in the `xmlmill.jar` file (location: **com/xmlmill/resources/**, but can be changed if necessary).*

## 8. Java API

This chapter describes the changes of in the **PDX** API in this release.

☞ *For a full overview of all methods, please visit the JavaDoc API in the **docs/apidocs/** directory in the download.*

### 8.1. Class: PDXLogFile

This class has been deprecated. Use **PDXLogger** class instead.

### 8.2. Class: PDXLogger

This class replaces the **PDXLogFile** class. As of this version the logging is based on the industry-standard **Log4j** package. **Log4j** allows the developer to control which log statements are output with arbitrary granularity. It is fully configurable at runtime using external configuration files. This implies that:

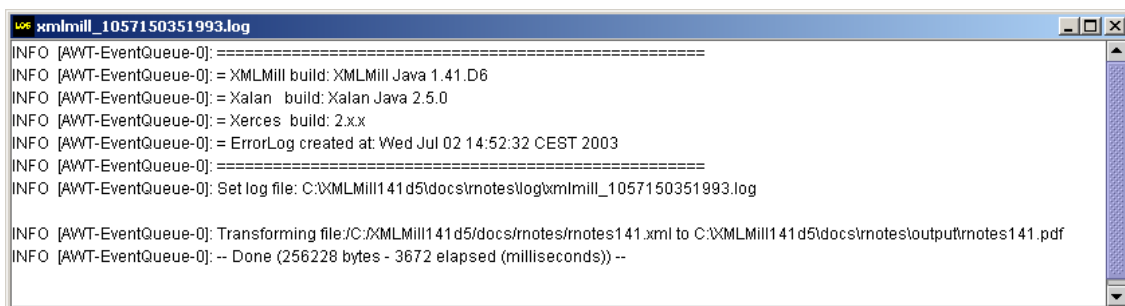
- ◆ XMLMill's logging behavior has been changed.
- ◆ New methods have been added to use your own **org.apache.log4j.Logger** instance.

#### 8.2.1. Default Logging behavior

If no logfile or logger instance is defined, the default logging will apply:

- ◆ All logmessages are send to the standard error stream (**System.err**).
- ◆ All logmessages with a level of at least **org.apache.log4j.Level.INFO** will logged.
- ◆ A default layout will be used.

A **.log** file could look like this:



```
xmlmill_1057150351993.log
INFO [AWT-EventQueue-0]: =====
INFO [AWT-EventQueue-0]: = XMLMill build: XMLMill Java 1.41.D6
INFO [AWT-EventQueue-0]: = Xalan build: Xalan Java 2.5.0
INFO [AWT-EventQueue-0]: = Xerces build: 2.xx
INFO [AWT-EventQueue-0]: = ErrorLog created at: Wed Jul 02 14:52:32 CEST 2003
INFO [AWT-EventQueue-0]: =====
INFO [AWT-EventQueue-0]: Set log file: C:\XMLMill141d5\docs\notes\log\xmlmill_1057150351993.log

INFO [AWT-EventQueue-0]: Transforming file:J:\XMLMill141d5\docs\notes\notes141.xml to C:\XMLMill141d5\docs\notes\output\notes141.pdf
INFO [AWT-EventQueue-0]: -- Done (256228 bytes - 3672 elapsed (milliseconds)) --
```

The default logging will contain following information:

1. The log level.
2. The thread-name.
3. The log message.

It is possible to overrule this default behavior by defining:

- ◆ The logfile to use (instead of the **System.err**) by using one of the four **setPDXLogFile(...)** methods.
- ◆ The loglevel by using the **setLogLevel(org.apache.log4j.Level)**.
- ◆ Passing a **org.apache.log4j.Logger** instance (containing a **org.apache.log4j.Appender** and **org.apache.log4j.Level**).

### 8.2.2. public static void setLevel(org.apache.log4j.Level )

Use this method to define which messages should be logged.

### 8.2.3. public static void setLogger(org.apache.log4j.Logger)

Use this method to define your own **logger** that should be used.

### 8.2.4. public static void resetLogger()

Reset the logger to the default logger:

- ◆ Send all messages to the **System.err**
- ◆ Set the loglevel at **org.apache.log4j.Level.INFO** level.
- ◆ Set the layout to the default **org.apache.log4j.PatternLayout**

## 8.3. Class: PDXTransform

The **PDXTransform** class has been modified to reflect the new logging system

### 8.3.1. setFeature(String, boolean)

The **http://com.xmlmill/transform/print-comments** feature has been suppressed. If comments need to be logged you need to set the **loglevel** to **DEBUG** using the **setLogLevel(Loglevel)** method in the **PDXLogger** class.

The **http://com.xmlmill/transform/print-status-messages** feature has been suppressed. If no status messages need to be logged you need to set the **loglevel** to **WARN** using the **setLogLevel(Loglevel)** method in the **PDXLogger** class.

## 8.4. Class: PDXTransformErrorHandler

The **PDXTransform** class has been modified to reflect the new logging system

### 8.4.1. Fields

- |               |  |
|---------------|--|
| PRINT_ERROR   | This field has been suppressed as whether or not error messages are logged depend on the <a href="#">loglevel</a> set. |
| PRINT_WARNING | This field has been suppressed as whether or not warning messages are logged   |

depend on the [loglevel](#) set.

`PRINT_STACKTRACE` This field has been suppressed as whether or not stacktraces are logged depend on the [loglevel](#) set.

#### **8.4.2. void PDXTransformErrorHandler()**

This method has been suppressed as, with the new logging system, whether or not warnings are logged depend on the [loglevel](#) set.

#### **8.4.3. boolean isPrintWarning()**

This method has been suppressed as, with the new logging system, whether or not warnings are logged depend on the [loglevel](#) set.

#### **8.4.4. boolean isPrintError()**

This method has been suppressed as, with the new logging system, whether or not errors are logged depend on the [loglevel](#) set.

#### **8.4.5. boolean isPrintStackTrace()**

This method has been suppressed as, with the new logging system, whether or not the **StackTrace** are logged depend on the [loglevel](#) set.

### **8.5. Class: Configurator**

The **Configurator** class defines which configuration file to use when transforming xml/xsl documents.

The **Configurator** is a static class that needs only (before the first transformation) to be called once during the life-time of your JVM session.

☞ *In a servlet environment it is best to call the **Cnfigurator** in the **init()** method of a servlet.*

#### **8.5.1. How to use the Configurator**

There are three scenario's how to define XMLMill's configuration:

1. Do not call the **Configurator**.
2. Call the **Configurator.configure()** method with no parameter.
3. Call the **Configurator.configure()** method passing a parameter indicating which configuration file to use.

##### **8.5.1.1. Do not call the Configurator**

When the **Configurator.configure()** method is not called, the built-in configuration options (defined in the `com/xmlmill/resources/config.xml` file) are used.

☞ *These values are the default values used in previous versions of XMLMill.*

#### 8.5.1.2. Call the **Configurator** with no parameter.

Call the **configure()** method (see below) without passing any parameter. XMLMill will read the values in the built-in **config.xml** file (located in the **com\xlmmill\conf** directory in the **xmlmill.jar** file).

☞ *The default values in the built-in **config.xml** file are the same as mentioned in the section [Do not call the Configurator](#), but can be changed if needed.*

#### 8.5.1.3. Call the **Configurator** passing a parameter

Call the **configure()** method passing a parameter indicating which configuration file to use. XMLMill will use the values defined in this configuration file.

#### 8.5.2. Method: **static synchronized void configure()**

Use this method if the default configuration file (**config.xml** in the **com\xlmmill\conf** directory) should be used.

#### 8.5.3. Method: **static synchronized void configure(String s)**

Use this method to pass a **java.lang.String** instance representing the configuration file as a URL. The URL should be fully resolved. For example: `http://www.xmlmill.com/test/config.xml` or `file:/c:/xmlmill/test/config.xml`.

#### 8.5.4. Method: **static synchronized void configure(URL url)**

Use this method to pass a **java.net.URL** instance representing the configuration file. The URL should be fully resolved. For example: `http://www.xmlmill.com/test/config.xml` or `file:/c:/xmlmill/test/config.xml`.

#### 8.5.5. Method: **static synchronized void configure(File f)**

Use this method to pass a **java.io.File** instance representing the configuration file. The **File** instance should be fully resolved. For example: `c:/xmlmill/test/config.xml`.

#### 8.5.6. Method: **static void setErrorHandler(PDXTransformErrorHandler handler)**

Define the errorhandler to use where all exceptions should be send to.

## 9. XMLMill batch

Some batch arguments are modified to reflect the new logging system.

### 9.1. Arguments

#### 9.1.1. Suppressed

- c The **printComments** argument has been suppressed. If comments need to be logged, the [loglevel](#) needs to be set to **debug**.

#### 9.1.2. New

- e Defines on which errors XMLMill should abort 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.
- g 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**.
- c Defines the configuration file to use.

xmlmill

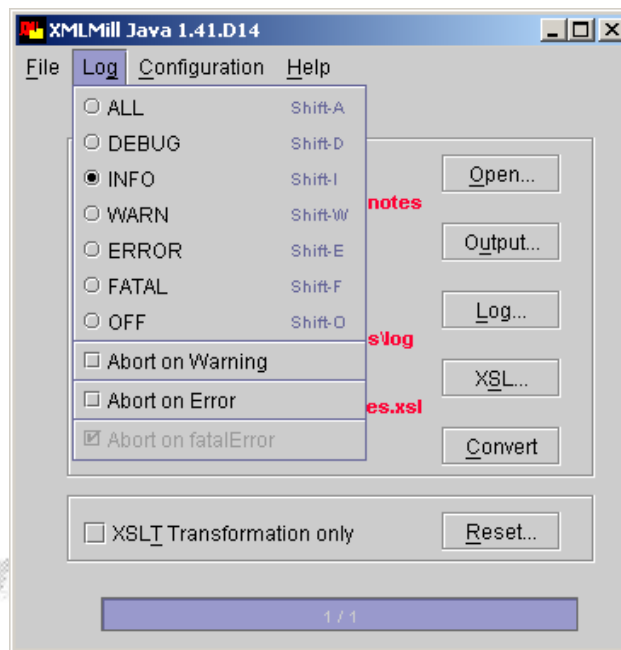
## 10. XMLMill Interactive

The graphical user-interface has been modified to reflect the new logging system and configuration feature.

### 10.1. Graphical User Interface

#### 10.1.1. Log

The **Log** menu defines the different options regarding the content of the **.log** file:



The different [loglevels](#) have been added to **Log** menu:

- ◆ OFF
- ◆ DEBUG
- ◆ INFO
- ◆ WARN
- ◆ ERROR
- ◆ FATAL
- ◆ ALL

The **Warnings** and **Errors** options have been suppressed

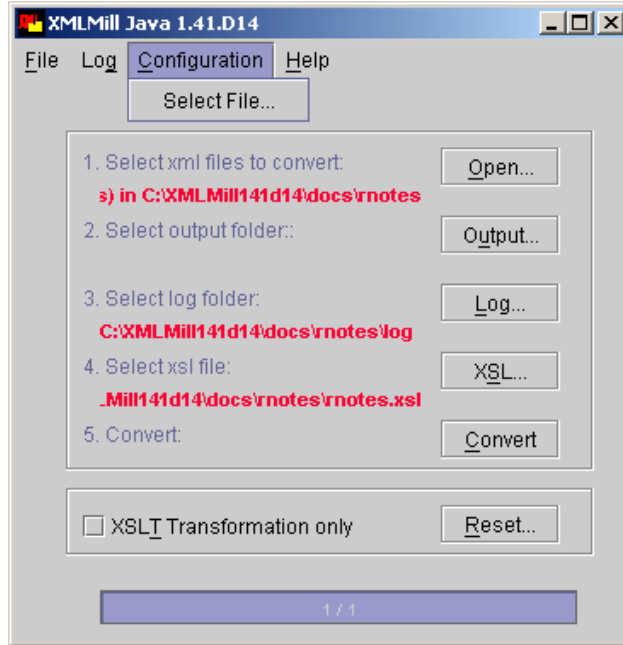
☞ *Set the **loglevel** respectively to **WARN** or **ERROR** in case you want to have these messages logged.*

The **Comments** and **printStackTrace** options have been suppressed

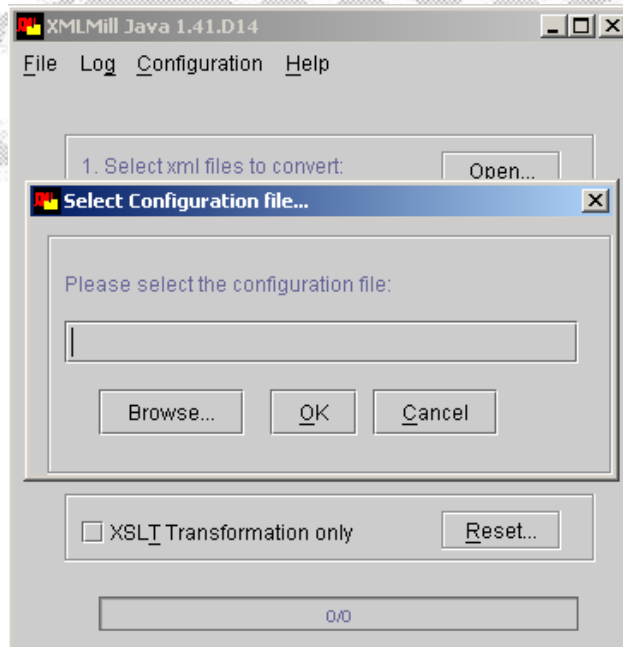
☞ Set the **loglevel** to **DEBUG** in case you want to have these messages logged.

### 10.1.2. Configuration

With the **Configuration** menu a configuration file can be selected:



Select the **Select File...** item to get following dialog box:



Enter the path and filename of the configuration file or use the **Browse** button to select a file using the **FileChooser** dialog box.

Making the field empty will make XMLMill use the built-in configuration file:



## 11. Disclaimer

This software redistributes parts of The Apache Software Foundation binary code, hence the disclaimer below:

The Apache Software License, Version 1.1

- \* Copyright (c) 1999 The Apache Software Foundation. All rights reserved.
- \*
- \* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- \*
- \* 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \*
- \* 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \*
- \* 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:
  - \* "This product includes software developed by the
  - \* Apache Software Foundation (<http://www.apache.org/>)."
  - \* Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
- \*
- \* 4. The names "Xalan" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).
- \*
- \* 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.
- \*
- \* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
- \* =====
- \*
- \* This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, Lotus Development Corporation., <http://www.lotus.com>. For more information on the Apache Software Foundation, please see [<http://www.apache.org/>](http://www.apache.org/).
- \*