

Release Notes

XMLMill for Java - XMLMill for Domino

2.0

xmlmill

© 2000-2004 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 1.60 (or later)	3
2.2. Upgrading from version 1.53 (or earlier)	3
3. What's new ?	4
3.1. XMLMill for Java	4
3.2. XMLMill for Domino	4
4. Upgrading from version 1.60 (or later)	5
4.1. Using the transform_to_2.00_*.xsl stylesheets	5
4.1.1. Introduction	5
4.1.2. Using the transform_to_2.00.cmd batch file	5
4.2. Using the transform_config_to_200.xsl stylesheet	6
4.2.1. Introduction	6
4.2.1.1. <pagetemplate> element	6
4.2.1.2. element	6
4.2.1.3. <font-info> element	6
4.2.2. Using the transform_config_to_200.cmd batch file	6
4.3. Behavioral changes	7
4.3.1. Height of header and footer no longer automatically defined	7
4.3.2. topleft attribute relative with respect to the region	7
4.3.3. <table-of-contents> element in it own <page-sequence>	7
4.3.4. The page margins defaulting to zero (0)	8
5. Upgrading from version 1.53 (or earlier)	9
5.1. Using the transform_to_160.xsl stylesheet	9
5.2. Using the transform_to_160.cmd batch file	9
6. Main changes	11
6.1. Result tree structure (main)	11
6.2. Page Layout	12
6.2.1. Example	12
6.3. ML Element summary	14
6.3.1. barcode	14
6.3.2. basic-link	14
6.3.3. break	14
6.3.4. conditional-page-master-reference	14
6.3.5. date	14
6.3.6. datetime	14
6.3.7. documents	14
6.3.8. document	14
6.3.9. encryption	14
6.3.10. external-graphic	14
6.3.11. file	14
6.3.12. flow	15
6.3.13. inline	15
6.3.14. layout-master-set	15
6.3.15. leader	15

6.3.16. list-block.	15
6.3.17. list-bullets.	15
6.3.18. list-item-text.	15
6.3.19. list-label.	15
6.3.20. list-label-format.	15
6.3.21. list-numbers.	15
6.3.22. marker.	15
6.3.23. meta-author.	15
6.3.24. meta-creationdate.	15
6.3.25. meta-creator.	15
6.3.26. meta-keywords.	15
6.3.27. meta-info.	15
6.3.28. meta-subjects.	15
6.3.29. meta-title.	16
6.3.30. navigation.	16
6.3.31. p.	16
6.3.32. textbox.	16
6.3.33. page-count.	16
6.3.34. page-number.	16
6.3.35. page-total.	16
6.3.36. page-sequence.	16
6.3.37. page-sequence-master.	16
6.3.38. region-after.	16
6.3.39. region-before.	16
6.3.40. region-body.	16
6.3.41. region-end.	16
6.3.42. region-start.	16
6.3.43. repeatablepage-master-alternatives.	16
6.3.44. repeatablepage-master-reference.	16
6.3.45. retrieve-marker.	17
6.3.46. simple-page-master.	17
6.3.47. single-page-master-reference.	17
6.3.48. static-content.	17
6.3.49. table.	17
6.3.50. table-and-caption.	17
6.3.51. table-of-contents.	17
6.3.52. table-body.	17
6.3.53. table-caption.	17
6.3.54. table-cell.	17
6.3.55. table-column.	17
6.3.56. table-footer.	17
6.3.57. table-header.	17
6.3.58. table-row.	17
6.3.59. time.	17
6.3.60. title.	18
6.3.61. toc-header.	18
6.3.62. toc-style.	18
6.3.63. viewerpreferences.	18

7. XMLMill Interactive. 19

8. XMLMill Batch. 20

9. XMLMill API changes. 21

10. XMLMill for Domino.	22
10.1. servlets.properties.	22
11. Appendix A: Result tree structure 1.61.	23
12. Appendix B: Result tree structure 2.00.	24
13. Appendix C: Table structure 1.61.	25
14. Appendix D: Table structure 2.00.	26
15. Appendix E: List-block structure 2.00.	27

xmlmill

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 **PDX** api.
- ◆ **userguide.pdf** -- XMLMill user's guide.
- ◆ **dtdguide.pdf** -- An explanation of XMLMill's elements (tags) and their attributes.

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

☞ *This document is completely generated with XMLMill 2.00 using **rnotes200.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 1.60 (or later)

A lot of new elements and attributes have been added to this version. You need to transform your existing .xml/.xsl documents to a 2.00 compatible format.

☞ Please visit the [Upgrading from version 1.60 \(or later\)](#) chapter in this document for more information.

2.2. Upgrading from version 1.53 (or earlier)

Version 1.60 corrected the wrong conversion rate internally used to convert inches (in), centimeters (cm) and millimeters (mm) to pixels.

If you migrate from version 1.53 or earlier you need to modify your units of measurement (cm, mm and in) in order keep the current presentation of your documents.

☞ Please visit the [Upgrading from version 1.53 \(or earlier\)](#) chapter in this document for more information.

You also need to apply other modifications (if necessary) from previous versions. The best procedure to make your transition as smooth as possible is as follows:

1. Back up all your xml/xsl/mill 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/xsl/mill files 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 2.00 of XMLMill. The main changes are:

General:

- ◆ A lot of effort has gone in moving the **m1:** namespace more into the direction of the **fo:** namespace standard (although the **m1:** namespace is kept), facilitating the transformation from the **fo:** namespace to the **m1:** namespace (for those willing to use the 'high volume xml/xsl to pdf generation' capability of XMLMill).
- ◆ Increased executing speed: between 10% and 30% (results based on comparing the volume tests from version 1.61 and 2.00).

Main Functionalities:

- ◆ Multiple page-templates possible (following the **fo:** specifications regarding Declarations and Pagination and Layout Formatting Objects).
- ◆ Apply conditionally page-templates.
- ◆ Page templates can be combined so different page-templates can be applied on different pages (for example odd-pages-, even-pages).
- ◆ Add markers in headers or footers.
- ◆ Page headers and footers can contain different elements like table(s), textbox(es), paragraphs, images, horizontal lines, ...
- ◆ Running headers and footers in a table possible.
- ◆ Ability to redefine the content of header/footers during table generation.
- ◆ A table can now contains a (repeating) table-footer and a caption (table-and-caption)
- ◆ Rotate textboxes 90, 180 or 270 degrees.

3.2. XMLMill for Domino

This release notes document describes all new (or enhanced) functionalities of version 1.60 of XMLMill. The main changes are:

1. See previous section.
2. See the Domino chapter.

4. Upgrading from version 1.60 (or later)

Two important changes have been done to make the new functionalities possible:

1. A lot of tags/attributes were renamed in order to move more closely to the **fo:** specification. As a result you need to transform your existing .mill/.xsl sheets to the 2.00 format.
2. The layout of the **config.xml** file has also changed. If you use an external **config.xml** or updated the internal **config.xml** (the one located in the **xmllmill.jar** file) you need to modify it so it is compliant with the new **config.dtd** definition (the default **config.xml** and **config.dtd** are located in the **conf/** directory in the download).

This chapter describes how to perform the conversions.

VERY IMPORTANT

☞ *PLEASE BACKUP ALL YOUR .MILL / .XSL FILES BEFORE STARTING THE TRANSFORMATION PROCESS.*

4.1. Using the transform to 2.00 *.xsl stylesheets

4.1.1. Introduction

To assist you in transforming your current .mill/xsl stylesheet(s) to a 2.00 compliant format a conversion stylesheet has been added to the download in the **etc/transform_to_200** directory.

Two versions exist:

- transform_to_200_with_ml_ns.xsl** This stylesheet will transform a .mill/.xsl sheet that uses the **ml:** namespace.
- transform_to_200_without_ml_ns.xsl** This stylesheet will transform a .mill/.xsl sheet that **does not** use the **ml:** namespace.

The **transform_to_200_*.xsl** stylesheet will automatically transform all obsolete tags/attributes to the new tags/attributes valid in the 2.00 version. The transformed sheet contains a lot of detailed information describing the changes done.

☞ *We encourage you to consult the converted document(s) and study closely the modifications done. This way you'll get to know the new 2.00 structure.*

4.1.2. Using the transform to 2.00.cmd batch file

In order to facilitate the conversion process a **.cmd** file has been added.

The default content is (.cmd):

```
@echo off
REM -----
REM -- Batch to convert .mill and .xsl stylesheets used in versions
REM -- prior to version 2.00 to level 2.00
REM -- More information please read the release notes in (rnotes200.pdf
REM -- in the docs/rnotes directory in the download.
REM -----

REM PLEASE CHANGE CLASSPATH TO INCLUDE JAR FILES CORRECTLY.
REM PLEASE CHANGE JAVA_HOME POINTING TO JAVA.EXE.
REM PLEASE CHANGE XMLMILL_HOME POINTING TO XMLMILL'S 'ROOT' DIRECTORY .

set XML_HOME=c:\xmllmill200
set JAVA_HOME= "C:\Program Files\JavaSoft\sdk131_07\bin\java"

set CLASSPATH=%XML_HOME%\lib\xercesimpl.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xalan.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xml-apis.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xmllmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xsltc.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\log4j.jar
```

```
REM -- Use basic transformer to convert .mill file
%JAVA_HOME% org.apache.xalan.xslt.Process -IN your_file.mill -XSL
transform_to_200_with_ml_ns.xsl -OUT your_file-200.mill

REM -- Use basic transformer to convert .xsl file
%JAVA_HOME% org.apache.xalan.xslt.Process -IN your_file.xsl -XSL
transform_to_200_with_ml_ns.xsl -OUT your_file-200.xsl
```

After defining the `xml_home`, `java_home` and `classpath` path variables a `org.apache.xalan.xslt.Process` can be called with 3 parameters:

- IN Refers to the .mill or .xsl file to transform.
- XSL Refers to the `transform_to_200_with_ml_ns.xsl` or `transform_to_200_without_ml_ns.xsl` stylesheet
- OUT Refers to the name of the transformed file

☞ *This way you can transform as well .mill as .xsl sheets to the 2.00 format.*

☞ *Please check this resulting sheet for any TODO action you need might to perform manually (Ctrl-F on 'TODO').*

☞ *In order to make yourself acquainted as quickly as possible with the new 2.00 standard, study the converted .mill or .xsl documents in order to see what changes has been done with your code.*

4.2. Using the transform config to 200.xsl stylesheet

4.2.1. Introduction

To assist you in transforming your current `config.xml` file to a 2.00 compliant format a conversion stylesheet has been added to the download in the `etc/transform_to_200` directory.

☞ *You only need to perform this transformation in case you modified the default `config.xml` in the `xmlmill.jar` file or used an external one.*

4.2.1.1. <pagetemplate> element

Following changes were applied:

- media** This attribute has been replaced with the corresponding **page-width** and **page-height** attributes.
- orientation** This attribute is no longer valid and has been discarded.
- font-weight** This attribute has been added.
- margin-top** The value of this attribute has been set to '0cm'
- margin-right** The value of this attribute has been set to '0cm'
- margin-bottom** The value of this attribute has been set to '0cm'
- margin-left** The value of this attribute has been set to '0cm'

Following changes are introduced:

4.2.1.2. element

- ◆ The **<generic>** section has been added to define a font for each 'generic' font.

4.2.1.3. <font-info> element

- ◆ The **font-style** attribute has been split in **font-style** and **font-weight** (CSS compliant).

4.2.2. Using the **transform config to 200.cmd** batch file

In order to facilitate the conversion process a **.cmd** file has been added.

The default content is (.cmd):

```
@echo off
REM -----
REM -- Batch to convert .mill and .xsl stylesheets used in versions
REM -- prior to version 2.00 to level 2.00
REM -- More information please read the release notes in (rnotes200.pdf
REM -- in the docs/rnotes directory in the download.
REM -----

REM PLEASE CHANGE CLASSPATH TO INCLUDE JAR FILES CORRECTLY.
REM PLEASE CHANGE JAVA_HOME POINTING TO JAVA.EXE.
REM PLEASE CHANGE XMLMILL_HOME POINTING TO XMLMILL'S 'ROOT' DIRECTORY .

set XML_HOME=c:\xmlmill200
set JAVA_HOME= "C:\Program Files\JavaSoft\jdk131_07\bin\java"

set CLASSPATH=%XML_HOME%\lib\xercesimpl.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xalan.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xml-apis.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xmlmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xsltc.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\log4j.jar

REM -- Use basic transformer to convert .mill file
%JAVA_HOME% org.apache.xalan.xslt.Process -IN your_config.xml -XSL
transform_config_to_200.xsl -OUT your_config-200.mill
```

After defining the **xml_home**, **java_home** and **classpath** path variables a **org.apache.xalan.xslt.Process** can be called with 3 parameters:

- IN Refers to the config.xml file to transform.
- XSL Refers to the **transform_to_config_200.xsl** stylesheet
- OUT Refers to the name of the transformed file

4.3. Behavioral changes

Please find below some changes in the behavior of the some elements:

4.3.1. Height of header and footer no longer automatically defined

In previous versions the header's and footer's height were automatically calculated based on its' content. As of this version the header and footer are defined using the **<region-before>** and **<region-after>** regions, which requires that the height is defined when the element is defined (using the **extent** attribute).

The **overflow** attribute defines if the content is **clipped** or made **visible** (allowing the content to 'spill' over to the next region).

The **transform_to_200_with_ml_ns.xsl** or **transform_to_200_without_ml_ns.xsl** stylesheets define te attribute as **visible**.

In case the region content spills over to the adjacent region, increase the value of the **extent** attribute.

4.3.2. **topleft** attribute relative with respect to the region

In previous versions the **topleft** values were with respect to the **page**. As of this version the values are with respect to the region the element it belongs to is contained in.

As a result you need to modify the topleft values if they contain an absolute value, like **topleft='3cm,2cm'**.

The transformed .xsl or .mill document indicates with what values the **topleft** attribute should be changed to.

4.3.3. **<table-of-contents>** element in it own **<page-sequence>**.

As of this version the `<table-of-contents>` element needs to be build in its own `<page-sequence>` so that the page-number and page-total can be correctly calculated. For an example check the `Element_table-of-contents.mill` example in the `samples/mill` directory in the download.

4.3.4. The page margins defaulting to zero (0).

As of this version the margins of the page default to zero. In previous versions the margins were defined to **2.19cm** in the default `config.xml` configuration file. These values have been set to **0cm**.

In case you do not use the transformation stylesheets to automatically transform you `.mill` or `.xsl` documents, you need to add the margin attributes (and correct values) to the `<ml:simple-page-master>` element.

```
Previous version (no margin attribute(s) defined):
...
<pagetemplate>
...
</pagetemplate>
...

2.00 version (margin attribute(s) should be explicitly defined):
...
<ml:simple-page-master margin="2.19cm">
...
</ml:simple-page-master>
...
```

If the margins are not explicitly defined in the `<ml:simple-page-master>` element, the values are defined by the default values as defined in the `config.xml` configuration file, meaning **0.cm** as of this version.

☞ *If you use the transformation sheets to do the conversion, the margin attributes are automatically added.*

5. Upgrading from version 1.53 (or earlier)

If you have a version 1.53.r4 or earlier you need to perform 2 transformations:

1. First the transformation described in this chapter
2. Then the transformation described in the [Upgrading from version 1.60 \(or later\)](#) chapter.

This chapter describes how to do the conversion.

5.1. Using the transform to 160.xsl stylesheet

It was discovered that the conversion of these measures to the correct number of pixels was based on the wrong conversion rate (in versions up to version 1.53.r4).

Please find below an overview of the conversions rates that were applied and the correct ones:

Unit of Measurement	Conversion rate applied	Correct Conversion rate
cm	1 cm = 25 pixels	1 cm = 28.3464567 pixels
mm	1 mm = 2.5 pixels	1 mm = 2.83464567 pixels
in	1 in = 96 pixels	1 in = 72 pixels
px	1 px = 1 pixel	1 px = 1 pixel

As you can see the conversion of **cm** and **mm** to pixels led to an underestimation of the number of pixels and the conversion of **in** to pixels led to an overestimation of the number of pixels.

As of version 1.60 the correct conversion rates are applied. As a result you will need to go through your **.mill** or **.xsl** document(s) and modify your measurement values accordingly.

To assist you a conversion stylesheet has been added to the download in the **etc/transform_to_160** directory.

You can use the **transform_to_160.xsl** stylesheet and the **transform.cmd** batch file to automatically convert your **.mill** or **.xsl** files.

The stylesheet will automatically convert the attributes where **cm**, **mm** or **in** are used according to following table:

Unit of Measurement	Correction rate
1 cm	0.881944444
1 mm	0.881944444
1 in	1.33333
1 px	1

As a result the layout of your documents will **not** be changed.

Examples:

If you have defined **margin="2.5cm"** then it should be changed to **margin="2.20cm"** (2.5 * 0.8819444).

If you have defined **padding="1in"** then it should be changed to **padding="1.33in"** (1 * 1.33).

5.2. Using the transform to 160.cmd batch file

In order to facilitate the conversion process a **.cmd** file has been added.

The default content is (.cmd):

```
@echo off
REM -----
```

```

REM -- Batch to convert .mill and .xsl stylesheets used in versions
REM -- prior to version 1.60 to level 1.60
REM -- More information please read the release notes in (rnotes200.pdf
REM -- in the docs/rnotes directory in the download.
REM -----

REM PLEASE CHANGE CLASSPATH TO INCLUDE JAR FILES CORRECTLY.
REM PLEASE CHANGE JAVA_HOME POINTING TO JAVA.EXE.
REM PLEASE CHANGE XMLMILL_HOME POINTING TO XMLMILL'S 'ROOT' DIRECTORY .

set XML_HOME=c:\xmlmill200
set JAVA_HOME= "C:\Program Files\JavaSoft\jdk131_07\bin\java"

set CLASSPATH=%XML_HOME%\lib\xercesimpl.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xalan.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xml-apis.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xmlmill.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\xsltc.jar
set CLASSPATH=%CLASSPATH%;%XML_HOME%\lib\log4j.jar

REM -- Use basic transformer to convert .mill file
%JAVA_HOME% org.apache.xalan.xslt.Process -IN your_file.mill -XSL transform_to_160.xsl
-OUT your_file-160.mill

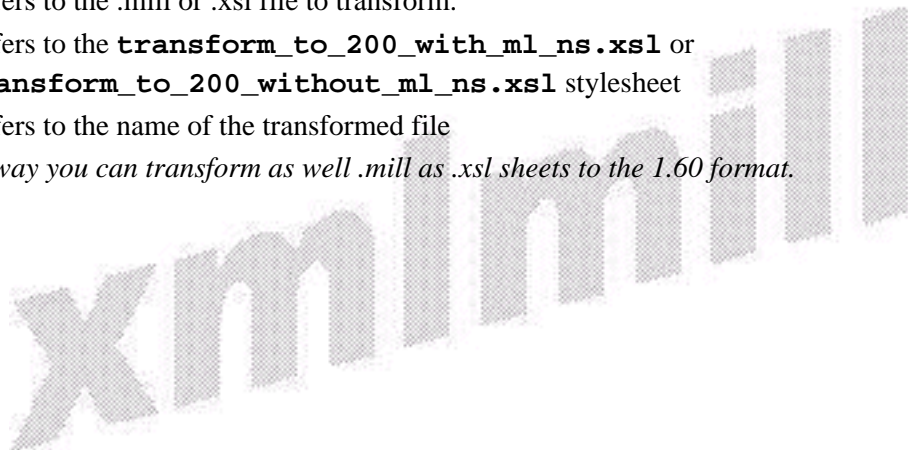
REM -- Use basic transformer to convert .xsl file
%JAVA_HOME% org.apache.xalan.xslt.Process -IN your_file.xsl -XSL transform_to_160.xsl
-OUT your_file-160.xsl

```

After defining the `xml_home`, `java_home` and `classpath` path variables a `org.apache.xalan.xslt.Process` can be called with 3 parameters:

- IN Refers to the .mill or .xsl file to transform.
- XSL Refers to the `transform_to_200_with_ml_ns.xsl` or `transform_to_200_without_ml_ns.xsl` stylesheet
- OUT Refers to the name of the transformed file

☞ *This way you can transform as well .mill as .xsl sheets to the 1.60 format.*



6. Main changes

6.1. Result tree structure (main)

In order to add new functionalities and move closer to the **fo**: specification a lot of tags were renamed or added. :

As a result the result tree structure has significantly changed.

Up to version 1.61 the result tree structure was:

☞ See [Appendix A: Result tree structure 1.61](#)

As of version 2.00 the new main structure is:

☞ See [Appendix B: Result tree structure 2.00](#)

Following changes were done to extend the functionality and are (partially) **fo**: compliant:

- ◆ The tag regarding the Pagination and Layout Formatting (**<ml:pagetemplate>**) has been replaced with a far more sophisticated element model (**ml:layout-master-set** and its descendants).
- ◆ The **<ml:header>** and **<ml:footer>** tags are replaced by **<static-content>** tags allowing to define any content in headers or footers (and no longer only text or an image).
- ◆ The **<ml:content>** tag has been replaced by one or more **<page-sequence>** tags. Each **<page-sequence>** has a **master-reference** attribute referring to a **<ml:simple-page-master>** or **<ml:page-sequence-master>** element, allowing to apply different layouts for different pages.
- ◆ A page is divided into five different regions: **<ml:region-before>**, **<ml:region-body>**, **<ml:region-start>**, **<ml:region-end>** and **<ml:region-after>**.
- ◆ The **<ml:static-content>** defines the content to be repeated on each page in a region (which region is defined by the **master-reference** attribute).
- ◆ The **<ml:flow>** attributes contains the elements to be used to generate the actual content of a page.
- ◆ The **<ml:table-and-contents>** is a wrapper around the **<ml:table>** element, allowing to define a caption for a table.
- ◆ The **<ml:table>** element has been extensively modified (new tag children) in order to provide more functionality (running headers and footers). See [Appendix D: Table structure 2.00](#)
- ◆ The **<ml:image>** element has been replaced with the **<external-graphic>** element.
- ◆ The **<ml:line>** element has been replaced with the **<ml:leader>** element.
- ◆ The **<ml:box>** attribute has been suppressed. As of now a box can be defined as an empty **<textbox>**.
- ◆ The **<ml:newpage>** attribute has been suppressed. A new page is generated based on the **break-before** or **break-after** attributes of some elements.

Other added functionalities:

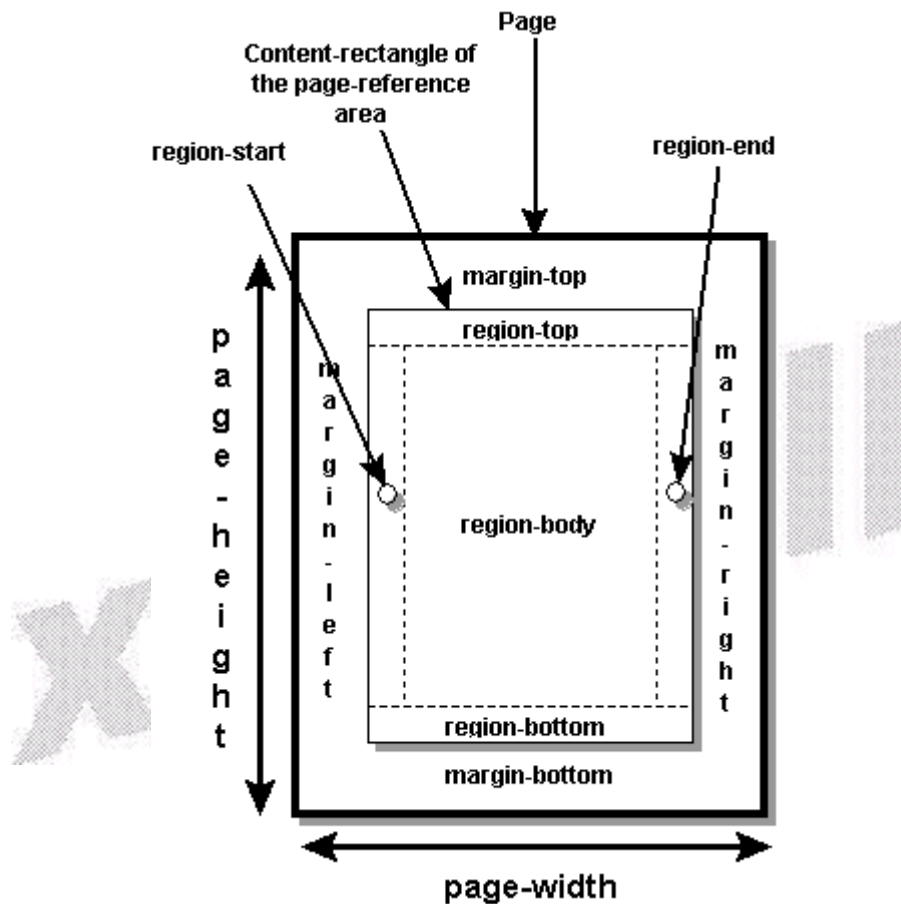
- ◆ The Code128 barcode has been added as barcode specification (**code128**).
- ◆ The Australian Post 'remittance advice' barcode has been added as barcode specification (**auspost1**).
- ◆ A barcode can be added in a **<table-cell>**.

- ◆ Textboxes can be rotated 0°, 90°, 180° or 270°.
- ◆ A JPG image can be in CMYK format.
- ◆ The batch mode has an extra parameter **-x**, used to define the minimum-size a pdf document should be.

6.2. Page Layout

The simple page-layout model (`<ml:page-template>`) has been replaced has been replaced with a far more sophisticated element model (`<ml:layout-master-set>` and its descendants).

As a result each page is divided into five regions (similar to the **fo:** specification) that can each be written to.



☞ The `<ml:layout-master-set>` and its descendants are fully **fo:** compliant. Please visit the [userguide.pdf](#) document for more information (or the [W3C website](#)).

6.2.1. Example

Below we compare a 1.61 .mill example and the converted 2.00 result:

```

1.61 .mill example
<?xml version="1.0" encoding="UTF-16"?>
<ml:documents 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 xmlmill.xsd">
  <ml:document file="output/article.pdf">
    <!-- Define the pagetemplate (the layout of the page) -->
    <ml:pagetemplate media="a4" orientation="portrait" margin="2cm"
      font="type1 timesroman normal 10pt">
      <ml:header align="left">
        <ml:image src="file:images/xmlmill.jpg" />
      </ml:header>
      <ml:header align="right"><ml:page-number /></ml:header>
    
```

```

    <ml:footer align="right"><ml:time/></ml:footer>
    <ml:footer align="left"><ml:file/></ml:footer>
  </ml:pagetemplate>
  <!-- Define the content (the layout of the page) -->
  <ml:content>
  ...

  2.00 .mill example
  <?xml version="1.0" encoding="UTF-16"?>
  <ml:documents 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 xmlmill.xsd">
  <ml:document file="output/article.pdf" font="normal 10pt Times">
  <!-- Change: Add <layout-master-set> -->
  <ml:layout-master-set> ❶
  <!-- Change: Add <simple-page-master> --> ❶
  <ml:simple-page-master master-name="internal" reference-orientation="0"
    page-width="595" page-height="842"
    margin-left="2cm" margin-right="2cm"
    margin-top="10pt" margin-bottom="10pt">
  <!-- Add <region-body> --> ❷
  <ml:region-body column-count="1" margin-bottom="2cm" margin-top="2cm"
    background-image="file:images/wmxmlmill.jpg"
    background-position-horizontal="center"
    background-position-vertical="center"/>
  <!-- Add <region-before> --> ❸
  <ml:region-before extent="2cm" precedence="true"/>
  <!-- Add <region-after> --> ❹
  <ml:region-after extent="2cm" precedence="true"/>
  </ml:simple-page-master>
  </ml:layout-master-set> -->
  <ml:page-sequence initial-page-number="1" master-reference="internal">❺
  <!-- Add <static-content> -->❻
  <ml:static-content flow-name="xsl-region-before">
  <ml:external-graphic src="file:images/xmlmill.jpg" align="left" valign="top"
  content-height="100%" content-width="100%"/>
  <ml:textbox valign="top" align="left" text-align="right" >Page
  <ml:page-number/> of <ml:page-total/> page(s)</ml:textbox>
  </ml:static-content>
  <ml:static-content flow-name="xsl-region-after">
  <ml:textbox valign="top" align="left" text-align="right">
  <ml:time/>
  </ml:textbox>
  <ml:textbox valign="top" align="left" text-align="left">
  <ml:file/>
  </ml:textbox>
  </ml:static-content>
  <!-- Add <flow> -->
  <ml:flow flow-name="xsl-region-body"> ❻
  <ml:external-graphic src="file:images/headerwaragainst.jpg" text-wrapping="on"
  content-height="100%" width="100%" content-width="scale-to-fit"/>
  ...

```

Above example shows the important changes done to the page layout and pagination of a .mill document:

- ❶ The **<pagetemplate>** is replaced by a **<simple-page-master>** element and its descendants. This element contains the attributes regarding page size, margins and orientation of the elements on the page.
- ❷ The **<region-body>** defines the region in which the page's contents is written. It defines a margin-top and margin-bottom to be sure the region-body does not interfere with the region-before and region-after. Also the background-image properties are defined here.
- ❸ The **<region-before>** element defines the region in which the page's header content is written. Compared with the previous version the region-before does not contain the actual content of the header (this is done in the static-content element).
- ❹ The **<region-after>** element defines the region in which the page's footer content is written. Compared with the previous version the region-after does not contain the actual content of the header (this is done in the static-content element).
- ❺ The content of a page and subsequent pages are defined with the **<page-sequence>** element and its descendants.
- ❻ The **<page-sequence>** element can contains multiple **<ml:static-content>** elements defining the content of a region (defined with the **flow-name** attribute). Here comes the content

of the `<header>` and `<footer>` elements of the previous version.

- ⑦ The `<page-sequence>` element can only contain one `<ml:flow>` element defining the content of a page(s). Here comes the content of the `<content>` element of the previous version.
 - ☞ *It is important to note that, in contradiction with the previous version, the height of the `<ml:region-before>` and `<ml:region-before>` elements (representing the header and footer) need to be 'manually' defined using the **extend** attribute. In previous version the header and footer's height were dynamically defined by the content of the header and footer.*
 - ☞ *In order to make yourself acquainted as quickly as possible with the new 2.00 standard, study the converted `.mill` or `.xsl` documents in order to see what changes has been done with your code.*

6.3. ML Element summary

Below a summary is given of all elements (tags) available in this version. For a complete description how to use each element please visit the [dtdguide.pdf](#) in the `docs/` directory in the download or online.

6.3.1. barcode

The `ml:barcode` element is used for representing the bars of a barcode type.

6.3.2. basic-link

The `ml:basic-link` element is used for representing the resource of a simple link.

6.3.3. break

The `ml:break` element is used for inserting a carriage-return in a certain position in a sentence.

6.3.4. conditional-page-master-reference

The `ml:conditional-page-master-reference` is used to identify a page-master that is to be used when the conditions on its use are satisfied

6.3.5. date

The `ml:date` element represents the current date.

6.3.6. datetime

The `ml:datetime` element represents the current date and current time.

6.3.7. documents

The `ml:documents` node is the top node of an XSL result tree.

6.3.8. document

The `ml:document` node is the starting node of a partial XSL result tree that represents all elements to generate a simple `.pdf` document (a XSL result tree can contain multiple `ml:document` elements) .

6.3.9. encryption

The `ml:encryption` element is used for defining the encryption parameters to encrypt the pdf document being generated.

6.3.10. external-graphic

The `ml:external-graphic` flow element is used for a graphic where the graphics data resides outside of the XML result tree in the `fo` namespace.

6.3.11. file

The ml:file element represents the file name (and directory) of the pdf document being generated.

6.3.12. flow

The content of the ml:flow element is a sequence of flow elements that provides the flowing text content that is distributed into pages.

6.3.13. inline

The ml:inline element is commonly used for formatting a portion of text with a background or enclosing it in a border.

6.3.14. layout-master-set

The ml:layout-master-set is a wrapper around all masters used in the document.

6.3.15. leader

The ml:leader element is used to generate leaders consisting either of a rule or of a row of a repeating character or cyclically repeating pattern of characters that may be used for connecting two text elements.

6.3.16. list-block

The ml:list-block element is used to format a list (of bullets or numbers).

6.3.17. list-bullets

The ml:list-bullets element is used to format a list of bullets.

6.3.18. list-item-text

The ml:list-item-text element represents the text in a list-numbers or list-bullets list.

6.3.19. list-label

The ml:list-label element defines the label to be used in bulleted list. This can be used to define the 'bullet-character' to use.

6.3.20. list-label-format

The ml:list-label-format element defines the format of the numbers in a list-numbers list.

6.3.21. list-numbers

The ml:list-bullets element is used to format a list of numbers.

6.3.22. marker

The ml:marker is used in conjunction with ml:retrieve-marker to produce running headers or footers.

6.3.23. meta-author

The ml:meta-author element defines the meta information regarding the author of the document.

6.3.24. meta-creationdate

The ml:meta-creationdate element defines the creation date of the document.

6.3.25. meta-creator

The ml:meta-creator element defines the creator(s) of the document.

6.3.26. meta-keywords

The ml:meta-keywords element defines the keyword(s) of the document.

6.3.27. meta-info

The ml:marker element acts as a holder for other meta elements.

6.3.28. meta-subjects

The ml:meta-title element defines the subject of the document.

6.3.29. meta-title

The ml:meta-title element defines the title of the document.

6.3.30. navigation

The navigation controls whether the document should contain a table of contents and/or outlines and defines the numbering format of the levels displayed in the table of contents and/or outlines.

6.3.31. p

The ml:p element is commonly used for formatting paragraphs, titles, headlines, etc.

6.3.32. textbox

The ml:textbox element is commonly used for formatting paragraphs, titles, headlines, etc.

6.3.33. page-count

The ml:page-count element is used to represent the total number of pages in the document.

6.3.34. page-number

The ml:page-number element is used to represent the current page-number.

6.3.35. page-total

The ml:page-total element is used to represent the total number of pages in a page-sequence.

6.3.36. page-sequence

The ml:page-sequence element is used to specify how to create a (sub-)sequence of pages within a document; for example, a chapter of a report. The content of these pages comes from flow children of the ml:page-sequence.

6.3.37. page-sequence-master

The ml:page-sequence-master specifies sequences of page-masters that are used when generating a sequence of pages.

6.3.38. region-after

This region defines a viewport that is located on the "after" side of ml:region-body region.

6.3.39. region-before

This region defines a viewport that is located on the "before" side of ml:region-body region.

6.3.40. region-body

This region specifies a viewport/reference pair that is located in the "center" of the ml:simple-page-master.

6.3.41. region-end

This region defines a viewport that is located on the "end" side of ml:region-body region.

6.3.42. region-start

This region defines a viewport that is located on the "start" side of ml:region-body region.

6.3.43. repeatablepage-master-alternatives

An ml:repeatablepage-master-alternatives specifies a sub-sequence consisting of repeated instances of a set of alternative page-masters. The number of repetitions may be bounded or potentially unbounded.

6.3.44. repeatablepage-master-reference

An ml:repeatablepage-master-reference specifies a sub-sequence consisting of repeated instances of a single page-master. The number of repetitions may be bounded or potentially unbounded.

6.3.45. retrieve-marker

The ml:retrieve-marker is used in conjunction with ml:marker to produce running headers or footers.

6.3.46. simple-page-master

The ml:simple-page-master is used in the generation of pages and specifies the geometry of the page. The page may be subdivided into up to five regions.

6.3.47. single-page-master-reference

An ml:single-page-master-reference specifies a sub-sequence consisting of a single instance of a single page-master.

6.3.48. static-content

The ml:static-content element holds a sequence or a tree of elements that is to be presented in a single region or repeated in like-named regions on one or more pages in the page-sequence. Its common use is for repeating or running headers and footers.

6.3.49. table

The ml:table flow element is used for formatting the tabular material of a table.

6.3.50. table-and-caption

The ml:table-and-caption flow element is used for formatting a table together with its caption.

6.3.51. table-of-contents

The ml:table-of-contents represents a table of contents that is automatically generated (in its own page-sequence).

6.3.52. table-body

The ml:table-body element is used to contain the content of the table body.

6.3.53. table-caption

The ml:table-caption element is used to contain block-level elements containing the caption for the table only when using the ml:tableand-caption.

6.3.54. table-cell

The ml:table-cell element is used to group content to be placed in a table cell.

6.3.55. table-column

The ml:table-column element specifies characteristics applicable to table cells that have the same column and span.

6.3.56. table-footer

The ml:table-footer element is used to contain the content of the table footer.

6.3.57. table-header

The ml:table-header element is used to contain the content of the table header.

6.3.58. table-row

The ml:table-row element is used to group table cells into rows.

6.3.59. time

The ml:time represents the current time.

6.3.60. title

The ml:title element is used to associate a title with a given page-sequence. This title may be used by an interactive User Agent to identify the pages. For example, the content of the ml:title can be formatted and displayed in a "title" window or in a "tool tip".

6.3.61. toc-header

The ml:toc-header element is used to generate the table-of-content caption title.

6.3.62. toc-style

The ml:toc-style element is used to define the format of the headings of different levels in a table-of-content.

6.3.63. viewerpreferences

The ml:viewerpreferences element defines the way the generated pdf document is to be displayed in Acrobat Reader (or compatible reader).

xmlmill

7. XMLMill Interactive

The interactive version of XMLMill has not changed.

xmlmill

8. XMLMill Batch

One parameter has been added:

- x Define the minimum-size (in bytes) a .pdf document should be.

xmlmill

9. XMLMill API changes

The API has not changed in this version.

xmlmill

10. XMLMill for Domino

10.1. `servlets.properties`

One additional parameter has been added to the `servlets.properties` file

```
# 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,
realpath=/lotus/domino/html/xmlmill

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

As of this version the `realpath` can be defined. This is an optional parameter that represents the absolute path to the 'xmlmill' directory (see the http://www.xmlmill.com/rnotes/installation_guide_r6.pdf online or in the directory `domino/docs` in the download).

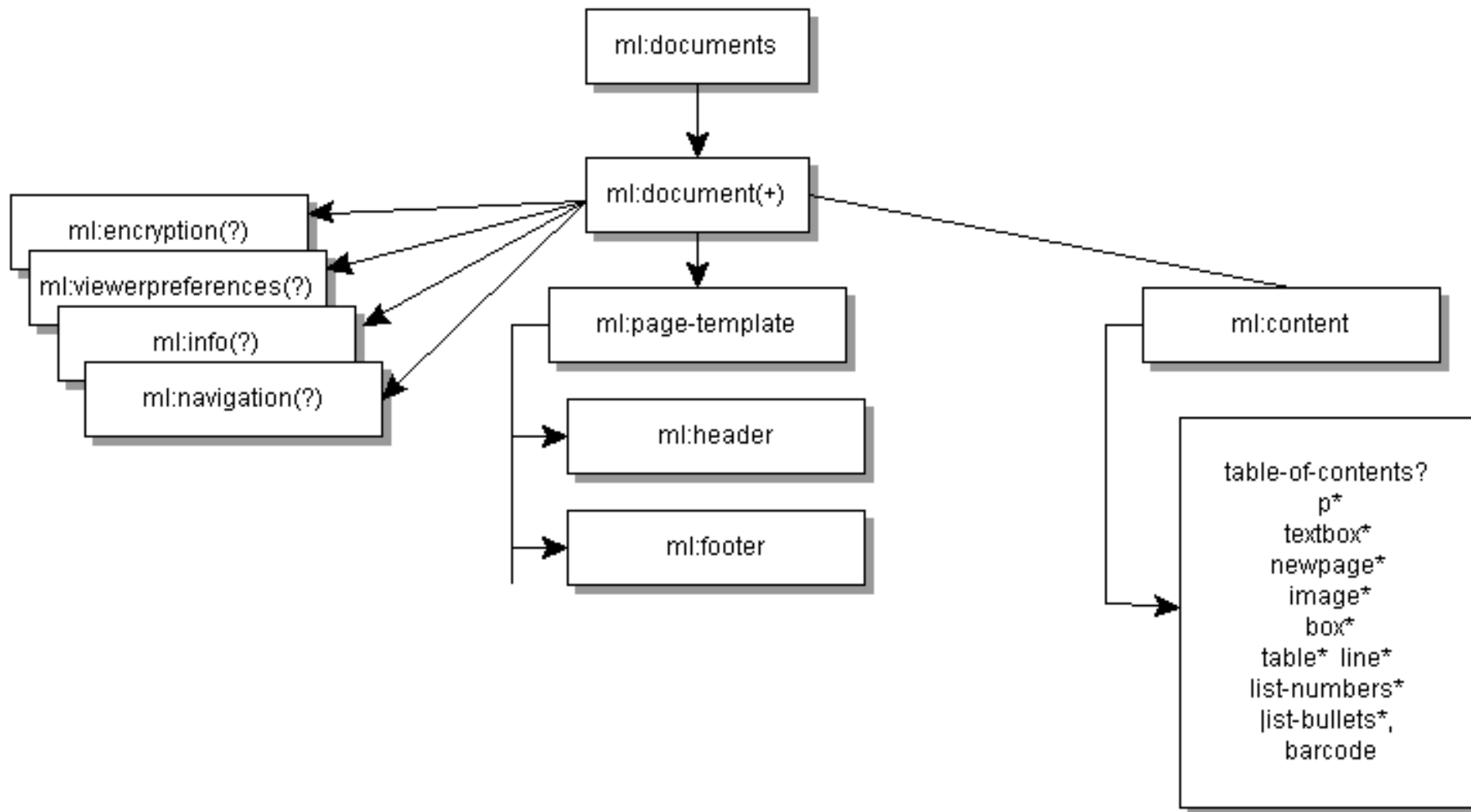
This parameter can be used in case the path to the xmlmill directory, which is normally automatically detected by XMLMill, cannot be found.

When this path cannot be found, the log directory will display something like this :

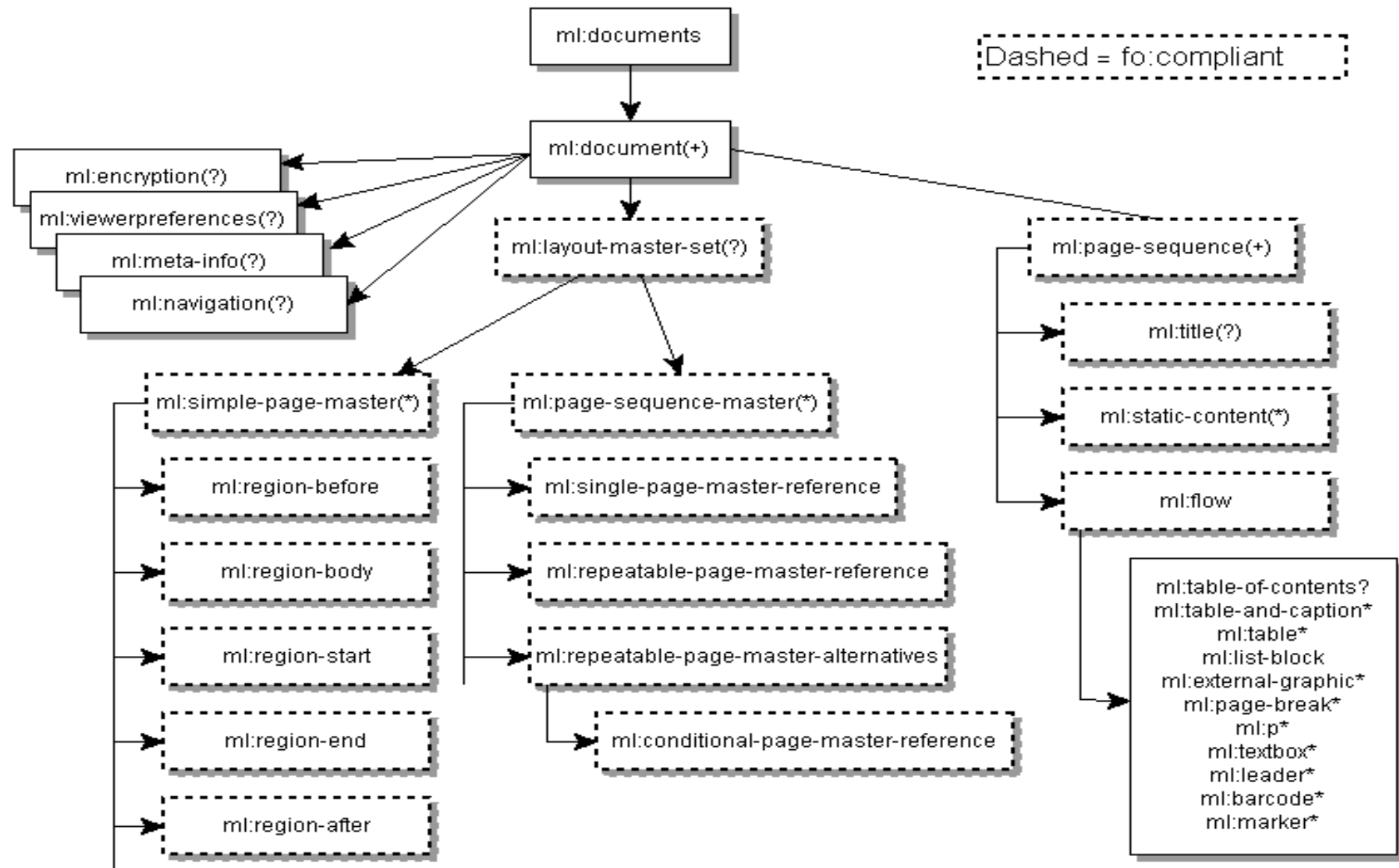
```
...
09/11/2004 08:47:48 HTTP JVM: INFO [Thread-115]: Realpath: null
...
09/11/2004 08:47:57 HTTP JVM: INFO [Thread-115]: Transforming with:
null/xsl/yourstylesheet.xsl
...
09/11/2004 08:47:57 HTTP JVM: INFO [Thread-115]:
Catch-error:javax.servlet.ServletException: xsl file not found
```

If this is the case, use the `realpath` parameter to define this path manually.

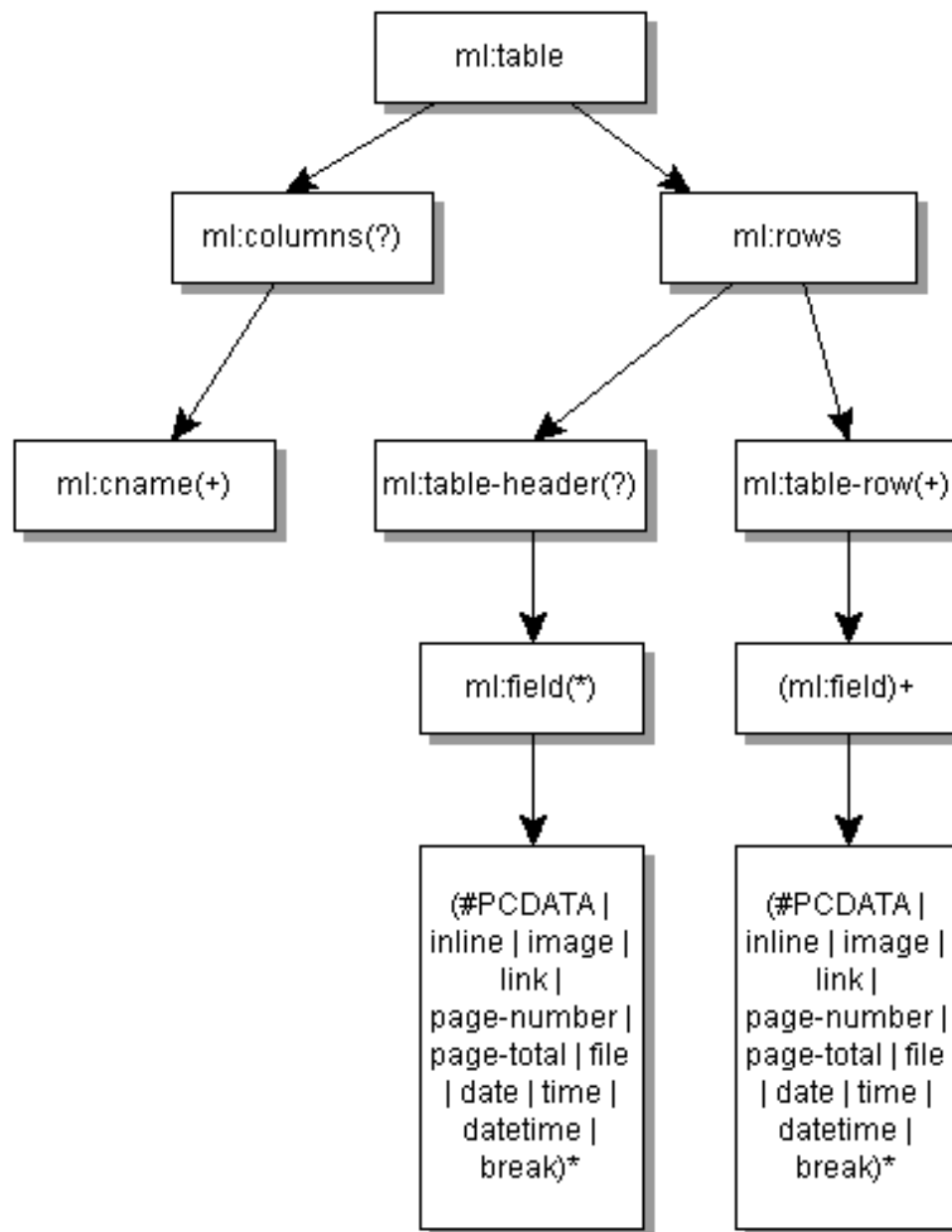
11. Appendix A: Result tree structure 1.61



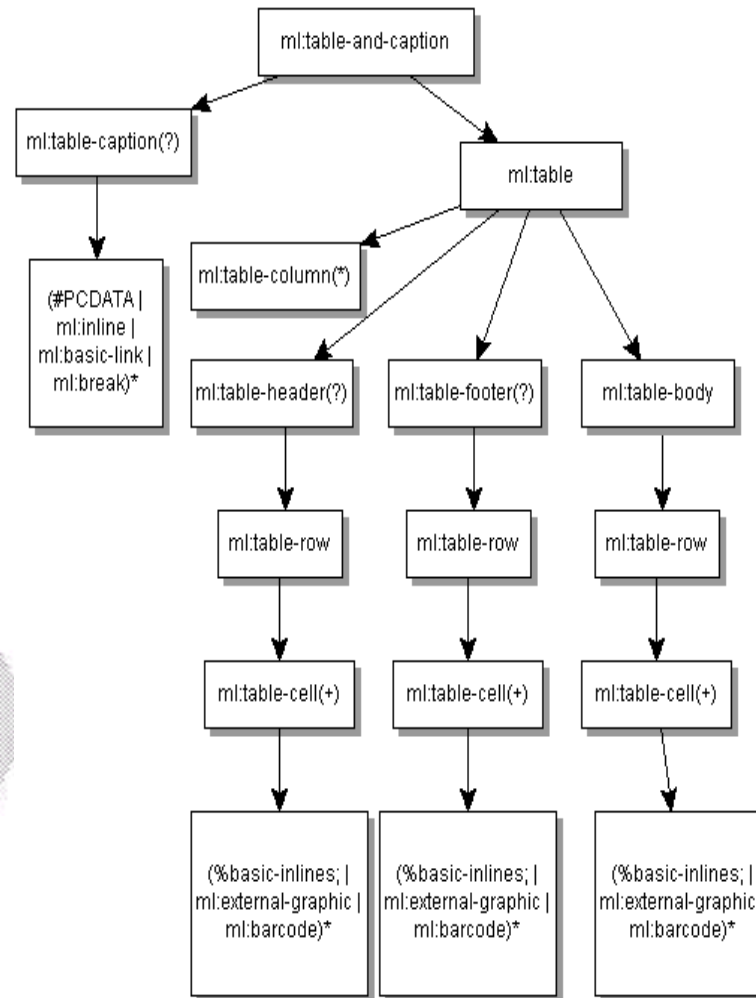
12. Appendix B: Result tree structure 2.00



13. Appendix C: Table structure 1.61



14. Appendix D: Table structure 2.00



15. Appendix E: List-block structure 2.00

