APT User Guide

Hussein Shafie,
Jean-Yves Belmonte,
Pixware
Immeuble Capricorne
23 rue Colbert
78180 Montigny Le Bretonneux
France
www.pixware.fr
hussein@pixware.fr
Phone: +33 (0)1 30 60 07 00
Fax: +33 (0)1 30 96 05 23

July 13, 2005

Contents

Chapter 1

Introduction

APT (Almost Plain Text) is a simple markup language (like HTML) than can be used to write simple article-like documents (like HTML). Unlike HTML, APT uses as few tags as possible to express the structure of the document. Instead, APT uses paragraph indentation.

The benefits of using APT are:

- APT documents are not tedious to type using text editors.
- When writing an APT document using a text editor, what you type is readable (i.e. not obfuscated by markup).
- APT documents can be embedded in source code (C, C++, Tcl) comments (a la javadoc).
- APT documents can be converted to many formats (currently LaTeX, PS, PDF, HTML, SGML or XML/DocBook, RTF) using aptconvert (lightweight, 100% Java, OpenSource tool).
- APT is content oriented and strictly structured. That's why APT documents are perennial. For example, it is possible to convert them to DocBook with almost no loss of structure.

The drawbacks of using APT are:

- You are limited to simple (simplistic?) article-like documents.
- The style of the documents generated by aptconvert is simple and neutral. It cannot be parametrized to enforce an entreprise specific look.
- Using a text editor to author structured documents is clearly not the wave of the future (which our current project *xmledit* is :-).

Chapter 2

The APT format

In the following section, boxes containing text in typewriter-like font are examples of APT source.

2.1 Document structure

A short APT document is contained in a single text file. A longer document may be contained in a ordered list of text files. For instance, first text file contains section 1, second text file contains section 2, and so on.

Note: Splitting the APT document in several text files on a section boundary is not mandatory. The split may occur anywhere. However doing so is recommended because a text file containing a section is by itself a valid APT document.

A file contains a sequence of paragraphs and “displays” (non paragraphs such as tables) separated by open lines.

A paragraph is simply a sequence of consecutive text lines.

```
  First line of first paragraph.
  Second line of first paragraph.
  Third line of first paragraph.

  Line 1 of paragraph 2 (separated from first paragraph by an open line).
  Line 2 of paragraph 2.
```

The indentation of the first line of a paragraph is the main method used by an APT processor to recognize the type of the paragraph. For example, a section title must not be indented at all.

A “plain” paragraph must be indented by a certain amount of space. For example, a plain paragraph which is not contained in a list may be indented by two spaces.

```
My section title (not indented).

  My paragraph first line (indented by 2 spaces).
```

Indentation is not rigid. Any amount of space will do. You don’t even need to use a consistent indentation all over your document. What really matters for an APT processor is whether the paragraph is not indented at all or, when inside a list, whether a paragraph is more or less indented than the first item of the list (more about this later).

```
    First paragraph has its first line indented by four
spaces. Then the author did even bother to indent the
other lines of the paragraph.

  Second paragraph contains several lines which are all
  indented by two spaces. This style is much nicer than
  the one used for the previous paragraph.
```

Note that tabs are expanded with a tab width set to 8.

2.2 Document elements

2.2.1 Block level elements

Title

A title is optional. If used, it must appear as the first block of the document.

```
                                   ------
                                   Title
                                   ------
                                   Author
                                   ------
                                    Date
```

A title block is indented (centering it is nicer). It begins with a line containing at least 3 dashes (`---`).

After the first `---` line, one or several consecutive lines of text (implicit line break after each line) specify the title of the document.

This text may immediately be followed by another `---` line and one or several consecutive lines of text which specifies the author of the document.

The author sub-block may optionaly be followed by a date sub-block using the same syntax.

The following example is used for a document with an title and a date but with no declared author.

```
                                   ------
                                   Title
                                   ------
                                   ------
                                    Date
                                   ------
```

The last line is ignored. It is just there to make the block nicer.

Paragraph

Paragraphs other than the title block may appear before the first section.

```
  Paragraph 1, line 1.
  Paragraph 1, line 2.

  Paragraph 2, line 1.
  Paragraph 2, line 2.
```

Paragraphs are indented. They have already been described in the document structure section.

Section

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

```
Section title

* Sub-section title

** Sub-sub-section title

*** Sub-sub-sub-section title

**** Sub-sub-sub-sub-section title
```

Section titles are not indented. A sub-section title begins with one asterisk (`*`), a sub-sub-section title begins with two asterisks (`**`), and so forth up to four sub-section levels.

List

```
      * List item 1.

      * List item 2.

        Paragraph contained in list item 2.

            * Sub-list item 1.

            * Sub-list item 2.

      * List item 3.
```

List items are indented and begin with a asterisk (`*`).

Plain paragraphs more indented than the first list item are nested in that list. Displays such as tables (not indented) are always nested in the current list.

To nest a list inside a list, indent its first item more than its parent list. To end a list, add a paragraph or list item less indented than the current list.

Section titles always end a list. Displays cannot end a list but the `[]` pseudo-element may be used to force the end of a list.

```
      * List item 3.
        Force end of list:

      []

--------------------------------------------
Verbatim text not contained in list item 3
--------------------------------------------
```

In the previous example, without the `[]`, the verbatim text (not indented as all displays) would have been contained in list item 3.

A single `[]` may be used to end several nested lists at the same time. The indentation of `[]` may be used to specify exactly which lists should be ended. Example:

```
      * List item 1.

      * List item 2.

            * Sub-list item 1.

            * Sub-list item 2.

            []

-------------------------------------------------------------------
Verbatim text contained in list item 2, but not in sub-list item 2
-------------------------------------------------------------------
```

There are three kind of lists, the bulleted lists we have already described, the numbered lists and the definition lists.

```
      [[1]] Numbered item 1.

                [[A]] Numbered item A.

                [[B]] Numbered item B.

      [[2]] Numbered item 2.
```

A numbered list item begins with a label beetween two square brackets. The label of the first item establishes the numbering scheme for the whole list:

`[[1]]`	Decimal numbering: 1, 2, 3, 4, etc.
`[[a]]`	Lower-alpha numbering: a, b, c, d, etc.
`[[A]]`	Upper-alpha numbering: A, B, C, D, etc.
`[[i]]`	Lower-roman numbering: i, ii, iii, iv, etc.
`[[I]]`	Upper-roman numbering: I, II, III, IV, etc.

The labels of the items other than the first one are ignored. It is recommended to take the time to type the correct label for each item in order to keep the APT source document readable.

```
      [Defined term 1] of definition list 2.

      [Defined term 2] of definition list 2.
```

A definition list item begins with a defined term: text between square brackets.

Verbatim text

```
----------------------------------------
Verbatim
         text,
                preformatted,
                                escaped.
----------------------------------------
```

A verbatim block is not indented. It begins with a non indented line containing at least 3 dashes (`---`). It ends with a similar line.

`+--` instead of `---` draws a box around verbatim text.

Like in HTML, verbatim text is preformatted. Unlike HTML, verbatim text is escaped: inside a verbatim display, markup is not interpreted by the APT processor.

Figure

```
[Figure name] Figure caption
```

A figure block is not indented. It begins with the figure name between square brackets. The figure name is optionally followed by some text: the figure caption.

The figure name is the pathname of the file containing the figure but without an extension. Example: if your figure is contained in `/home/joe/docs/mylogo.jpeg`, the figure name is `/home/joe/docs/mylogo`.

If the figure name comes from a relative pathname (recommended practice) rather than from an absolute pathname, this relative pathname is taken to be relative to the directory of the current APT document (a la HTML) rather than relative to the current working directory.

Why not leave the file extension in the figure name? This is better explained by an example. You need to convert an APT document to PostScript and your figure name is `/home/joe/docs/mylogo`. A APT processor will first try to load `/home/joe/docs/mylogo.eps`. When the desired format is not found, a APT processor tries to convert one of the existing formats. In our example, the APT processor tries to convert `/home/joe/docs/mylogo.jpeg` to encapsulated PostScript.

Table

A table block is not indented. It begins with a non indented line containing an asterisk and at least 2 dashes (`*--`). It ends with a similar line.

The first line is not only used to recognize a table but also to specify column justification. In the following example,

- the second asterisk (`*`) is used to specify that column 1 is centered,
- the plus sign (`+`) specifies that column 2 is left aligned,
- the colon (`:`) specifies that column 3 is right aligned.

```
*----------*--------------+----------------:
| Centered | Left-aligned | Right-aligned  |
| cell 1,1 | cell 1,2     | cell 1,3       |
*----------*--------------+----------------:
| cell 2,1 | cell 2,2     | cell 2,3       |
*----------*--------------+----------------:
Table caption
```

Rows are separated by a non indented line beginning with `*--`.

An optional table caption (non indented text) may immediately follow the table.

Rows may contain single line or multiple line cells. Each line of cell text is separated from the adjacent cell by the pipe character (`|`). (`|` may be used in the cell text if quoted: `\|`.)

The last `|` is only used to make the table nicer. The first `|` is not only used to make the table nicer, but also to specify that a grid is to be drawn around table cells.

The following example shows a simple table with no grid and no caption.

```
*----*------*
 cell | cell
*----*------*
 cell | cell
*----*------*
```

Horizontal rule

```
=====================
```

A non indented line containing at least 3 equal signs (===).

Page break

```
^L
```

A non indented line containing a single form feed character (Control-L).

2.2.2 Text level elements

Font

```
  <Italic> font. <<Bold>> font. <<<Monospaced>>> font.
```

Text between < and > must be rendered in italic. Text between << and >> must be rendered in bold. Text between <<< and >>> must be rendered using a monospaced, typewriter-like font.

Font elements may appear anywhere except inside other font elements.

It is not recommended to use font elements inside titles, section titles, links and defined terms because a APT processor automatically applies appropriate font styles to these elements.

Anchor and link

```
  {Anchor}. Link to {{anchor}}. Link to {{http://www.pixware.fr}}.
  Link to {{{anchor}showing alternate text}}.
  Link to {{{http://www.pixware.fr}Pixware home page}}.
```

Text between curly braces (`{}`) specifies an anchor. Text between double curly braces (`{{}}`) specifies a link.

It is an error to create a link element that does not refer to an anchor of the same name. The name of an anchor/link is its text with all non alphanumeric characters stripped.

This rule does not apply to links to *external* anchors. Text beginning with `http:/`, `https:/`, `ftp:/`, `file:/`, `mailto:`, `../`, `./` (`..\` and `.\` on Windows) is recognized as an external anchor name.

When the construct **{{{***name***}***text***}}** is used, the link text *text* may differ from the link name *name*.

Anchor/link elements may appear anywhere except inside other anchor/link elements.

Section titles are implicitly defined anchors.

Line break

```
  Force line\
  break.
```

A backslash character (\) followed by a newline character.

Line breaks must not be used inside titles and tables (which are line oriented blocks with implicit line breaks).

Non breaking space

```
  Non\ breaking\ space.
```

A backslash character (\) followed by a space character.

Special character

```
  Escaped special characters: \~, \=, \-, \+, \*, \[, \], \<, \>, \{, \}, \\.
```

In certain contexts, these characters have a special meaning and therefore must be escaped if needed as is. They are escaped by adding a backslash in front of them. The backslash may itself be escaped by adding another backslash in front of it.

Note that an asterisk, for example, needs to be escaped only if its begins a paragraph. (`*` has no special meaning in the middle of a paragraph.)

```
  Copyright symbol: \251, \xA9, \u00a9.
```

Latin-1 characters (whatever is the encoding of the APT document) may be specified by their codes using a backslash followed by one to three octal digits or by using the `\x`*NN* notation, where *NN* are two hexadecimal digits.

Unicode characters may be specified by their codes using the `\u`*NNNN* notation, where *NNNN* are four hexadecimal digits.

Comment

```
~~Commented out.
```

Text found after two tildes (~~) is ignored up to the end of line.

A line of ~ is often used to “underline” section titles in order to make them stand out of other paragraphs.

2.3 The APT format at a glance

```
                                    ------
                                    Title
                                    ------
                                    Author
                                    ------
                                     Date

  Paragraph 1, line 1.
  Paragraph 1, line 2.

  Paragraph 2, line 1.
  Paragraph 2, line 2.

Section title

* Sub-section title

** Sub-sub-section title

*** Sub-sub-sub-section title

**** Sub-sub-sub-sub-section title

      * List item 1.

      * List item 2.

        Paragraph contained in list item 2.

            * Sub-list item 1.

            * Sub-list item 2.

      * List item 3.
        Force end of list:

      []

+------------------------------------------+
Verbatim text not contained in list item 3
+------------------------------------------+

      [[1]] Numbered item 1.

                [[A]] Numbered item A.

                [[B]] Numbered item B.

      [[2]] Numbered item 2.

  List numbering schemes: [[1]], [[a]], [[A]], [[i]], [[I]].

      [Defined term 1] of definition list.

      [Defined term 2] of definition list.
```

```
+-------------------------------+
Verbatim text
                        in a box
+-------------------------------+

  --- instead of +-- suppresses the box around verbatim text.

[Figure name] Figure caption

*----------*--------------+----------------:
| Centered | Left-aligned | Right-aligned  |
| cell 1,1 | cell 1,2     | cell 1,3       |
*----------*--------------+----------------:
| cell 2,1 | cell 2,2     | cell 2,3       |
*----------*--------------+----------------:
Table caption

  No grid, no caption:

*-----*------*
 cell | cell
*-----*------*
 cell | cell
*-----*------*

  Horizontal line:

=======================================================================

^L
  New page.

  <Italic> font. <<Bold>> font. <<<Monospaced>>> font.

  {Anchor}. Link to {{anchor}}. Link to {{http://www.pixware.fr}}.
  Link to {{{anchor}showing alternate text}}.
  Link to {{{http://www.pixware.fr}Pixware home page}}.

  Force line\
  break.

  Non\ breaking\ space.

  Escaped special characters: \~, \=, \-, \+, \*, \[, \], \<, \>, \{, \}, \\.

  Copyright symbol: \251, \xA9, \u00a9.

~~Commented out.
```

Chapter 3

Installing aptconvert

3.1 Install on Unix

Prerequisites:

- Java 1.3 or above (aptconvert is no longer tested using a Java 1.1 runtime).
- If you need to convert APT documents to PostScript, a recent TeX distribution such as teTeX 0.9 or MiKTeX 1.20e.

  TeX must support new option **--interaction=***mode* which is used to set TeX's interaction mode to batchmode.

  A number of non standard LaTeX packages such as *fancyhdr* are needed. See section about the LaTeX output format.
- If you need to convert APT documents to PDF, the TeX distribution plus a recent Ghostscript distribution (such as 5.10) for its `ps2pdf` utility.

Procedure:

1. Unpack the APT distribution somewhere. This requires approximately 2Mb of disk space.

   ```
   $ cd
   $ gzip -d -c apt.tgz | tar xvf -
   $ ls apt
   class
   distrib
   java
   makefile
   ```

   The binaries and the documentation are found in the distrib sub-directory. Class, java and makefile are only needed if you want to rebuild aptconvert.
2. Copy aptconvert and aptall.jar to a directory referenced in your path (example `/usr/local/bin`).

   ```
   $ su
   $ cp ~/apt/distrib/aptconvert /usr/local/bin
   $ cp ~/apt/distrib/aptall.jar /usr/local/bin
   $ chmod a+rx /usr/local/bin/aptconvert
   $ chmod a+r /usr/local/bin/aptall.jar
   ```

3. You're done unless you want to provide a system-wide configuration to aptconvert users.

   In this case,

   (a) Use a text editor to create a file containing aptconvert properties (more about this in the using aptconvert section). For example, create a file called `/usr/local/lib/aptconvert.rc`.

   (b) Use a text editor to modify last line of the script `/usr/local/bin/aptconvert`. Replace `java` by `java -Daptconvertrc=/usr/local/bin/aptconvert.rc`.

3.2 Install on Windows NT

Install on Windows NT is similar to the install on Unix but under Windows you'll have to copy distrib\aptconvert.bat rather than distrib/aptconvert.

Chapter 4

Using aptconvert

4.1 Converting APT documents to other formats

An aptconvert command looks like:
aptconvert *option ... option output_file input_file ... input_file*.

For example, the document you are currently reading, the APT User Guide, is contained in 4 input files: intro.txt, format.txt, install.txt and using.txt. Converting it to HTML is as simple as running
`aptconvert userguide.html intro.txt format.txt install.txt using.txt`.

Converting the format section (a valid APT document on its own) to PDF can be done by executing
`aptconvert format.pdf format.txt`.

4.1.1 Options

-v Print external commands (latex, ps2pdf, graphics conversion on the fly, etc) being executed .

Default: not verbose.

-toc Insert a table of content at the beginning of the generated document. Not supported by all output formats.

Default: no table of contents.

-index Insert a simple index at the end of the generated document. Not supported by all output formats.

APT anchors are used as index entries.

Default: no index.

-nonum Do not number sections.

Default: sections are numbered.

-meta *meta_key meta_value* Add meta-information named *meta_key* with value *meta_value* to the output document. Not supported by all output formats.

Example: `-meta keywords "XML XSL XSLT XPath"`.

Default: no metas.

-pi *format pi_key pi_value* Specify a **P**rocessing **I**nstruction named *pi_key* with value *pi_value*, to be used when generating format *format*.

Example: `-pi html homeURL http://www.pixware.fr`.

Default: no PIs.

-rule *src_ext dst_ext src_to_dst_rule* Specify a graphics conversion rule: execute command *src_to_dst_rule* to convert graphics file whose extension is *src_ext* to graphics file whose extension is *dst_ext*.

Example: `-rule fig jpg 'fig2dev -L jpeg %F %G'`

Src_to_dst_rule is a command template where all occurences of `%F` are substituted with the source graphics file name and where all occurences of `%G` are substituted with the destination graphics file name.

Default: no rules.

-paper *paper* Specifies paper size. Not all sizes are supported by all output formats.

Default: a4.

-lang *language* Specifies the document language. For example, this is used to translate “Contents” to “Table des matières” if *language* is **fr**. Not all languages are supported by all output formats.

Default: en.

-enc *encoding* Specifies the encoding of the document. Not all encodings are supported by all output formats.

Default: your platform default encoding.

-? *ext* Print info about converter or extractor associated to file extension *ext*.

A converter is a backend which translates an APT document to another format. A converter is associated to a file name extension. For example, the LaTeX converter is automatically used when the output file ends with `.tex`.

An extractor is a preprocessor which extracts an APT document embedded into source code (i.e. à la javadoc). A extractor is associated to a file name extension. For example, the Tcl extractor is automatically used when the input file ends with `.tcl`.

4.1.2 Configuration files

Options which are common to all the APT documents processed on your site (-lang, -paper, -rule) should be put in a system-wide configuration file rather than being passed to the command line. See installing aptconvert.

An aptconvert configuration file is a text file (a Java property file) containing lines in the form *property=value*.

Example:

```
v=1

toc=1

lang=fr

paper=a4

latex.usepackage.0=bookman

# Screen resolution is 110dpi not 72dpi (72/120=0.65).
.gif.eps=giftopnm %F | pnmtops -scale 0.65 -rle > %G
.fig.jpg=fig2dev -L jpeg %F %G

keywords=pixware realtime \
data acquisition
```

Blank lines are allowed. Lines beginning with `#` are ignored. A `\` at the end of each line must be used if *value* is more than one line long.

Properties corresponding to PIs are named *format***.***pi_key* and properties corresponding to rules are named **.***src_ext***.***dst_ext*. Other properties not containing dots and not recognized as options are assumed to be metas.

Each time aptconvert is run, it attempts to read options from:

1. The configuration file specified using the `aptconvertrc` Java property (i.e. `java -Daptconvertrc=/usr/lib/aptconvert.rc`).
2. The configuration file found in the `$HOME` user directory named `.aptconvert` on Unix and `aptconvert.ini` on Windows.
3. The command line.

4.1.3 APT figures and graphics files

This example describes how aptconvert handles figures. What follows are the actions taken by aptconvert when converting to HTML an APT document containing a figure named `images/architecture`:

1. HTML needs GIF or JPEG images. (PNG is not yet well supported by browsers.)

   Therefore aptconvert tries to access a file named *input_file_directory*/images/architecture.gif. If it finds such file, it copies it to *output_file_directory* (unless input and output file directories are the same) and that's it.

2. If it doesn't find such file, it tries to copy *input_file_directory*/images/architecture.jpeg.
3. If it doesn't find such file, it tries to copy *input_file_directory*/images/architecture.jpg.
4. Let's say that steps 1, 2 and 3 have failed but that, wisely enough, the aptconvert user has defined three graphics conversion rules:

   ```
   .gif.eps=giftopnm %F | pnmtops -scale 0.65 -rle > %G
   .fig.eps=fig2dev -L ps %F %G
   .fig.jpg=fig2dev -L jpeg %F %G
   ```

   Aptconvert tries to access a file named *input_file_directory*/images/architecture.gif because first rule begins with `.gif`, but this fails like step 1.

5. Aptconvert succeeds to access a file named *input_file_directory*/images/architecture.fig (because second and third rules begin with `.fig`).

   Therefore it attempts to find a rule converting Fig graphics to GIF, but this fails.

6. It finds a rule that may be used to convert Fig graphics to JPEG.

   Aptconvert runs the `fig2dev` graphics converter with appropriate arguments and the generated HTML document finally gets its image.

4.2 Output formats

4.2.1 HTML

Extensions	html htm
-toc	yes
-index	yes
-paper	N/A
-lang	en es de fr it
-enc	ASCII ISO8859_1 ISO8859_2 ISO8859_3 ISO8859_4 ISO8859_5 ISO8859_6 ISO8859_7 ISO8859_8 ISO8859_9 SJIS UTF8 UTF16 Cp1250 Cp1251 Cp1252 Cp1253 MacArabic MacCentralEurope MacCroatian MacCyrillic MacGreek MacHebrew MacIceland MacRoman MacRomania MacThai MacTurkish MacUkraine
Graphics formats	gif jpeg png

Processing instructions:

-pi html paging *level* Create an HTML page for each section whose level is *level* (1 to 5, 0 means no paging).

When paging is enabled, a navigation toolbar is automatically added at the top of generated HTML pages.

When paging is enabled, the output file name is used as a template to give each page its own file name. Example: `aptconvert -pi html paging 2 -toc -index userguide.html intro.txt format.txt install.txt using.txt` generates files named: userguide1.html, userguide2.html, userguide2_1.html, ..., userguidetoc.html, userguideindex.html.

Default: 0 (single-page document).

-pi html css *file_name* Copy file named *file_name* to output directory. Add a corresponding style sheet link to generated HTML pages. In XML mode, also add a corresponding <?xml-stylesheet?> processing instruction.

Default: no style sheet.

-pi html homeURL *URL* Add a home icon to navigation toolbar pointing to URL *URL*.

Default: no home icon.

-pi html xml yes|no xml=yes means generate XHTML (XML mode), xml=no means generate plain HTML (SGML mode).

Default: if the output file name ends with `.xhtml` or `.xhtm`, the default value is yes, otherwise it is no.

Note that the generated XHTML conforms to the compatibility guidelines recommended by the W3C and is therefore well supported by most browsers.

-pi html systemId *URL|file* Specify the system identifier of the DTD.

Default: none in SGML mode; http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
or http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
in XML mode (valid XML documents must have a systemId).

-pi html strict yes|no strict=yes means use the strict DTD rather than the transitional DTD.

When using the transitional DTD, some deprecated elements and attributes (such as center, border, compact, etc) well supported by old browsers are generated.

When using the strict DTD, these deprecated elements and attributes are replaced by equivalent CSS style attributes.

Default: no in SGML mode; yes in XML mode.

-pi html showTitle yes|no showTitle=yes means that a visible division containing the title, author and date of the document (specified in the APT document or specified using `-meta`) must be generated in the first HTML page.

if showTitle=no, the title, author and date of the document are just converted to (invisible) head/title and head/meta HTML elements.

Default: no.

Meta-information:

any Any meta found by aptconvert is added to all generated HTML pages.

4.2.2 LaTeX

Extensions	tex
-toc	yes
-index	yes
-paper	a4 a5 b5 letter legal executive
-lang	br ca hr cs da nl en eo et fi fr gl de el he hu ga it no pl pt ro ru gd sk sl es sv tr cy
-enc	ASCII Cp1250 Cp1252 Cp437 Cp850 Cp852 Cp865 ISO8859_1 ISO8859_2 ISO8859_3 ISO8859_4 ISO8859_5 MacRoman
Graphics formats	eps

Processing instructions:

-pi latex documentclass *class* Specify LaTeX document class: article or report. The class name may be preceeded by options specified using the LaTeX syntax.

Example: `-pi latex documentclass [11pt]report`.

Default: [*P*]article, *P* specified by -paper.

-pi latex usepackage.*N package* Specify LaTeX packages augmenting or replacing packages used by default. The package name may be preceeded by options specified using the LaTeX syntax.

N varies for 0 to 10, so there are 11 such PIs available.

Example: `-pi latex usepackage.5 [french]babel`.

Default packages:

- a4wide if -paper a4, unless classic=yes.
- fancyhdr if needed, see pagestyle.
- [*L*]babel, *L* specified by -lang.
- graphics.
- times, unless classic=yes.

- [T1]fontenc.
- [*E*]inputenc, *E* specified by -enc.

-pi latex pagestyle *style* Specify page style (plain, empty, headings, etc).

Default: plain if classic=yes or if the document has no title, otherwise a custom style using the fancyhdr package.

-pi latex hyphenation.*N* *list* Specify how to hyphenate some words. *list* is a list of “pre-hyphenated” words separated by spaces.

N varies for 0 to 10, so there are 11 such PIs available.

Example: `-pi latex hyphenation.0 "gno-mon gno-mons gno-mon-ly"`.

Default: builtin TeX hyphenation rules.

-pi latex resizegraphics yes|no If resizegraphics is set to yes, graphics too large are automatically shrinked to accommodate the page size.

Default: no.

-pi latex classic yes|no classic=yes means: do not attempt to improve LaTeX classic look by using PostScript fonts, using vertical space rather than indentation to separate paragraphs, etc.

Default: no (not the classic look!).

Meta-information:

author If the document begins with a title block and if the author sub-block is not explicitely specified, an author sub-block is created from the value of the **author** meta, if any.

Therefore it is possible to specify once for all that you are the author of all the documents your write by adding, for example, the following lines into your .aptconvertrc or aptconvert.ini:

```
author=Joe User\n\
juser@incredible-widgets.com
```

4.2.3 PostScript

The APT document is first converted to LaTeX before being converted to PostScript using the commands specified by **pass1** and **pass2**. Therefore everything said about LaTeX is relevant when outputting PostScript.

Extension is ps.

Processing instructions:

-pi ps pass1 *latex_to_dvi_command_template* Specify the command template used to convert LaTeX to DVI.

Default: `latex doc`.

-pi ps pass2 *dvi_to_ps_command_template* Specify the command template used to convert DVI to PS.

Default: `dvips -o %O doc`.

About command templates:

- The passes work as follows:

  1. A temporary directory is created.

2. This temporary directory becomes the current directory.
3. The APT input files are translated to file `./doc.tex`.
4. *Pass1* is run as many times as needed.
5. *Pass2*, if specified, is run one time.
6. *Pass3*, if specified, is run one time.
7. The temporary directory and all its content is destroyed.

- These variables are substituted in order to turn the template into an actual command:

  %T The absolute path of the temporary directory created to contain all the files needed by latex.

  %O The absolute path of the output file name.

- An empty string may be used to suppress a pass.

4.2.4 PDF

The APT document is first converted to LaTeX before being converted to PDF using the commands specified by **pass1**, **pass2**, and **pass3**. Therefore everything said about LaTeX is relevant when outputting PDF.

Extension is pdf.

-pi pdf pass1 ***latex_to_dvi_command_template*** Specify the command template used to convert LaTeX to DVI.

Default: `latex doc`.

-pi pdf pass2 ***dvi_to_ps_command_template*** Specify the command template used to convert DVI to PS.

Default: `dvips -o doc.ps doc`.

-pi pdf pass3 ***ps_to_pdf_command_template*** Specify the command template used to convert PS to PDF.

Default: `ps2pdf doc.ps %O`.

Example 1: add this to your .aptconvert/aptconvert.ini if you want an hypertext TOC and/or Index.

```
latex.usepackage.0=hyperref

pdf.pass1=latex doc
pdf.pass2=dvips -z -o doc.ps doc
pdf.pass3=ps2pdf doc.ps %O
```

Example 2: add this to your .aptconvert/aptconvert.ini if you prefer to use pdflatex (you still have a problem with graphics because pdflatex does not support EPS). Note how pass3 has been suppressed.

```
pdf.pass1=pdflatex doc
pdf.pass2=mv doc.pdf %O
pdf.pass3=
```

4.2.5 DocBook

Extensions	sgml (DocBook/SGML) xml (DocBook/XML)
-toc	N/A
-index	N/A
-paper	N/A
-lang	any (not checked)
-enc	ASCII ISO8859_1 ISO8859_2 ISO8859_3 ISO8859_4 ISO8859_5 ISO8859_6 ISO8859_7 ISO8859_8 ISO8859_9 SJIS UTF8 UTF16 Cp1250 Cp1251 Cp1252 Cp1253 MacArabic MacCentralEurope MacCroatian MacCyrillic MacGreek MacHebrew MacIceland MacRoman MacRomania MacThai MacTurkish MacUkraine
Graphics formats	gif jpeg eps

Processing instructions:

-pi docbook publicId *publicId* Specify the public identifier of the DTD.

Default: -//OASIS//DTD DocBook V4.1//EN in SGML mode; -//OASIS//DTD DocBook XML V4.0//EN in XML mode.

-pi docbook systemId *URL|file* Specify the system identifier of the DTD.

Default: none in SGML mode; http://www.oasis-open.org/docbook/xml/4.0/docbookx.dtd in XML mode (valid XML documents must have a systemId).

-pi docbook css *file_name* XML mode only. Copy file named *file_name* to output directory. Add a corresponding <?xml-stylesheet?> processing instruction to the generated document.

Default: no style sheet.

-pi docbook italic *elemTag* Specify the starting tag of the DocBook element to be generated when an APT italic element is found.

Default: <emphasis>.

-pi docbook bold *elemTag* Specify the starting tag of the DocBook element to be generated when an APT bold element is found.

Default: <emphasis role="bold">.

-pi docbook monospaced *elemTag* Specify the starting tag of the DocBook element to be generated when an APT monospaced element is found.

Default: <literal>.

-pi docbook horizontalRule *string* Specify the string (generally a comment) to be generated when an APT horizontalRule element is found.

Default: <!-- HR -->.

-pi docbook pageBreak *string* Specify the string (generally a comment) to be generated when an APT pageBreak element is found.

Default: <!-- PB -->.

-pi docbook lineBreak *string* Specify the string (generally a comment) to be generated when an APT lineBreak element is found.

Default: <!-- LB -->.

Meta-information: not yet implemented.

4.2.6 RTF

Extensions	rtf
-toc	no
-index	no
-paper	a3 a4 a5 b4 b5 executive ledger legal letter tabloid <w>x<h>
-lang	no
-enc	ASCII Cp1250 Cp1251 Cp1252 ISO8859_1
Graphics formats	ppm

Notes:

- Hypertext links are not supported.
- While table of contents generation is not directly supported, special paragraph styles are associated with section headers. MS-Word users may take advantage of these styles to generate a table of contents.
- Input PPM images are converted to BMP and embedded in the output file. Thus output RTF documents are self-contained, i. e. they do not contain any reference to graphics files.

Processing instructions:

-pi rtf topmargin *margin* Specify the top margin.
Unit: cm. Default: 2.

-pi rtf bottommargin *margin* Specify the bottom margin.
Unit: cm. Default: 2.

-pi rtf leftmargin *margin* Specify the left margin.
Unit: cm. Default: 2.

-pi rtf rightmargin *margin* Specify the right margin.
Unit: cm. Default: 2.

-pi rtf fontsize *size* Specify the base font size.
Unit: pts. Default: 10.

-pi rtf spacing *spacing* Specify the base vertical spacing. This controls the space between display blocks.
Unit: pts. Default: 10.

-pi rtf resolution *resolution* Specify the screen resolution. This determines image dimensions.
Unit: dpi. Default: 72.

-pi rtf imagetype palette|rgb Specify the image type. Use *rgb* for images with more than 256 colors.
Default: *palette*

-pi rtf imagedataformat ascii|raw Specify the image data format.
Default: *ascii*

Meta-information: none.

4.3 Embedding APT documents into source code

An APT document may be embedded into source code files. In fact, APT was originally designed to speed up the authoring of reference manuals for C++ class libraries by embedding documentation into the header files.

When given an input file which ends with the right extension, for example `.h` for a C header file, aptconvert will automatically try to extract APT document fragments out of it.

4.3.1 Tcl

The input file must end with `.tcl`.

Single line comment containing a document fragment:

```
#x  eof - Check for end of file condition on channel
```

is extracted and rendered as:

eof - Check for end of file condition on channel

Note that comment line begins with `#x` (`x` like e**X**tract), followed by two spaces because the document fragment is a paragraph.

Multi-line comment block containing a document fragment:

```
    #x
    #=====================
    #
    #  <<eof>> <channelId>
    #
```

is extracted and rendered as:

---

eof *channelId*

First comment line contains exclusively the `#x` marker. Following lines begin with a single `#`. Note that indentation of the comment block containing the document fragment is allowed.

An open line after single or multi-line document fragments is mandatory. Otherwise the source code found immediatly after a fragment is extracted to and put into a verbatim display.

Example:

```
#x
#
#  Returns  1 if an end of file condition occurred during the
#  most recent input operation on channelId (such as gets), 0
#  otherwise.
#
proc eof {channelId} {
     return [eofImpl $channelId]
}
```

is extracted and rendered as:

```
proc eof {channelId} {
     return [eofImpl $channelId]
}
```

Returns 1 if an end of file condition occurred during the most recent input operation on channelId (such as gets), 0 otherwise.

By default the verbatim display is inserted before the document fragment. It is possible to specify another place by using the `~~x` extraction directive.

Example:

```
#x
#===
#
#~~x
#
#  Returns  1 if an end of file condition occurred during the
#  most recent input operation on channelId (such as gets), 0
#  otherwise.
#
proc eof {channelId} {
     return [eofImpl $channelId]
}
```

is extracted and rendered as:

```
proc eof {channelId} {
     return [eofImpl $channelId]
}
```

Returns 1 if an end of file condition occurred during the most recent input operation on channelId (such as gets), 0 otherwise.

A box is drawn around extracted verbatim displays by using the `tcl.verbatim` PI:

-pi tcl verbatim plain|box Default: plain.

4.3.2 C/C++

Document extraction from C/C++ source file works exactly like extraction from Tcl files. The main difference is that C/C++ comments do not begin #.

The input file must end with `.c`, `.h`, `.cpp`, `.hpp`, `.cxx`, `.hxx`, `.cc` or `.hh`.

The PI used to draw a box around extracted verbatim displays is `c.verbatim`.

Examples:

```
//x  Single line paragraph.

    //x
    //  First line.
    //  Second line.
    //

/*x Single line paragraph. */

/*x
  First line.
  Second line.
*/

    /** Single line paragraph. */

    /**
     *  Returns the value of the attribute called <name>
     *  if such attribute exists or null otherwise.
     */
    virtual const xstAnyValue* FindAttribute(const char* name);
```

Notes:

- A comment block using the `/**` marker and then using a `*` at the beginning of each line may be indented.
- A comment block using the `/*x` marker and then using no special character at the beginning of each line may not be indented.
- A multi-line comment block using the `/**` or `/*x` marker must end with a line beginning with `*/`.

Chapter 5

Enhancements and bug fixes

5.1 July 13, 2005

HTML

- Enhancement suggested by Jesper L. Nielsen (in the form of a patch): `-pi html showTitle yes|no` allows to generate a visible division containing the title, author and date of the document.

  Previously (and also until now, unless `-pi html showTitle yes` has been specified), this information was just converted to (invisible) head/title and head/meta HTML elements.

5.2 July 01, 2005

TeX, PS, PDF

- Applied the patch sent by Jesper L. Nielsen which allows to convert an UTF-8 APT file to LaTeX and hence PostScript and PDF.

5.3 June 01, 2004

- Fixed the following bug:

  ```
  aptconvert -pi html css test.css test.html test.txt
  ```

  When running the above command, if `test.css` already exists in current directory, it is overwritten by an empty file.

5.4 April 30, 2004

- Changed license from LGPL to MIT.

5.5 January 05, 2004

- Fixed a bug that prevented using non breaking spaces (\) inside table cells.

5.6 September 22, 2003

HTML

- `-toc` with `-pi html paging 0` adds an “Up” icon (link to the TOC) to section titles for sections at level 1, 2, 3. Previously, this was only done for sections at level 1.
- Slightly changed the `apt_toc.gif` icon.

5.7 May 21, 2003

- When using Java[tm] 1.4+ aptconvert supports both “historical” encoding names such as ISO8859_1 and canonical encoding names such as “ISO-8859-1”.

5.8 March 24, 2003

- The distribution now includes the APT source of the documentation. This source is a real world example of an APT document.

  Another real world example is *XMLmind XML editor - User’s Guide*. Download it from http://www.xmlmind.com/xmledit

5.9 February 17, 2003

- Bug fix: what was specified using `-pi docbook pageBreak` was ignored (for example: `-pi docbook pageBreak "<?pagebreak?>"`). Instead aptconvert output `<!-- HR -->`.

5.10 September 3, 2002

- Changed license from GPL to LGPL.
- Fixed bugs related to location of graphics files for the HTML/XHTML and DocBook converters. Before this fix, it was not possible to put graphics file in a subdirectory of the directory containing the APT document (example: `doc/doc.txt` with all images in `doc/images/`).

5.11 March 11, 2002

- Added support for MacRoman and ISO8859_4 to the LaTeX/PS/PDF converter. Using these encodings probably requires having a very recent TeX distribution installed on your machine.
- Removed useless `xmlns="http://www.w3.org/1999/xhtml"` (fixed attribute) in generated XHTML.

5.12 December 19, 2001

- Added support for Macintosh encodings (MacRoman, etc) to HTML and Docbook.

5.13 August 14, 2001

- Bug fix: when generating LaTeX (or PostScript or PDF), the default page header (i.e the one using package fancyhdr) did not contain a properly escaped title. Reported by Hergen Harnisch.

5.14 May 18, 2001

APT format

- Added support for numbered lists. List numbering schemes are : [[1]] (decimal), [[a]] (lower-aplha) , [[A]] (upper-alpha), [[i]] (lower-roman), [[I]] (upper-roman).

RTF

- New output format.

HTML

- The HTML converter nows supports 4 subtly different HTML variants:
  - HTML as an “SGML application” using the transitional 4.0.1 DTD. (This is the default HTML variant.)
  - HTML using the strict 4.0.1 DTD. (When **-pi html strict yes** is used.)
  - XHTML 1.0 (HTML as an “XML application”) using the strict DTD. (When **-pi html xml yes** is used or simply when the output file name ends with `.xhtml` or with `.xhtm`.)
  - XHTML 1.0 using the transitional DTD. (When **-pi html xml yes** is used or when the output file name ends with `.xhtml?` in conjunction with **-pi html strict no**.)
- Slightly improved rendering.
- Generated HTML is much less (human) readable because I removed ’\n’ after some elements (ex. no ’\n’ after a </p>). (This extra whitespace used to disturb my XML editor.)
- Bug fix: if a document contains an author element and if an author meta is specified as a command line option or in the .aptconvert file, two <meta name="author"> were added to the generated HTML pages.

PS

- Added **-pi ps pass1** *latex_to_dvi_command_template*.
  Added **-pi ps pass2** *dvi_to_ps_command_template*.

PDF

- Added **-pi pdf pass1** *latex_to_dvi_command_template*.
  Added **-pi pdf pass2** *dvi_to_ps_command_template*.
  Added **-pi pdf pass3** *ps_to_pdf_command_template*.

  With these PIs it is now possible to use pdflatex or dvipdfm rather than dvips+ps2pdf to generate PDF.

DocBook

- Changed the generated tags to conform with Norman Walsh's Simplified DocBook XML DTD + <anchor>.
- Gave up the support of DocBook V3 (i.e. no <artheader>, etc).
- Renamed option publicID (resp. systemID) to publicId (resp. systemId).
- Added **-pi docbook css** *file_name*.

5.15 February 26, 2001

- **-pi docbook publicID** default is `-//OASIS//DTD DocBook V4.1//EN`.

  In XML mode, **-pi docbook publicID** default is `-//OASIS//DTD DocBook XML V4.1.2//EN`.

  Validated the SGML and the XML generated by aptconvert using James Clark's sp.

5.16 June 17, 2000

- Bug fix: when generating PostScript or PDF, LaTeX was not run enough times and the table of contents was sometimes wrong.
- Removed the misplaced and redundant `standalone='no'` declaration in DocBook/XML. Validated the XML generated by aptconvert using a patched Xerces-J 1.1.0 parser.

  Note that specifying **-pi docbook systemID** *URL|file* is required in order to generate valid DocBook/XML documents.

5.17 June 13, 2000

- Added support for DocBook/SGML (tested using James Clark's Jade and Norman Walsh's Modular DocBook Stylesheets) and DocBook/XML (not really tested due to the lack of robust XML tools).
- Enhanced aptconvert.bat to be able to pass it more than 9 arguments and to support the `TEMP` and `HOME` environment variables (if these are defined).

5.18 April 5, 2000

- When the construct **{{{***name***}***text***}}** is used, the link text *text* may differ from the link name *name*.
- Inside a link, text beginning with `http:/`, `https:/`, `ftp:/`, `file:/`, `mailto:`, `../`, `./` (`..\` and `.\` on Windows) is recognized as an external anchor name.

5.19 February 28, 2000

- Added **-pi latex resizegraphics yes|no**.

  If latex.resizegraphics is set to yes, graphics too large are automatically shrinked to accommodate the page size. Default: no.

- Added support for the **author** meta to the LaTeX converter.

  If the document begins with a title block and if the author sub-block is not explicitely specified, an author sub-block is created from the value of the **author** meta, if any.

  Therefore it is possible to specify once for all that you are the author of all the documents your write by adding, for example, the following lines into your .aptconvertrc or aptconvert.ini:

  ```
  author=Joe User\n\
  juser@incredible-widgets.com
  ```

- Aptconvert now runs latex in batchmode. Previously, errors found by latex caused aptconvert to block: latex was waiting for the user to type something.

  Latex is not supposed to find errors but this sometimes happens with special characters such as \0 (in fact, the author intended to type `\\0`) or the "power of two" Latin-1 character (a bug in the inputenc package?).

- The following encodings are now supported by the HTML converter: Cp1250, Cp1251, Cp1252, Cp1253.

5.20 February 15, 2000

First release.