pyhwp Documentation

Release 0.1b9.dev0

mete0r

Jul 21, 2017

Contents

Contents:

CHAPTER 1

pyhwp

HWP Document Format v5 parser & processor.

Features

- Analyze and extract internal streams out from a HWP Document Format v5 file
- (*Experimental*) Conversion to OpenDocument format (.odt) or plain text (.txt)

Installation

from pypi:

```
virtualenv pyhwp
pyhwp/bin/pip install --pre pyhwp  # Install pyhwp into a virtualenv directory
```

Or:

```
pip install --user --pre pyhwp  # Install pyhwp into user's home directory
```

Requirements

- CPython 2.5, 2.6, 2.7, Jython 2.5.3 or PyPy 2.0.2
- setuptools
- pycrypto (optional, to decode distribution docs)

Documentation & Development

- Documentation: http://pythonhosted.org/pyhwp/ [한국/조선어] [develop branch]
- Distribution: http://pypi.python.org/pypi/pyhwp

- Development: https://github.com/mete0r/pyhwp
- Issue tracker: https://github.com/mete0r/pyhwp/issues
- Feedbacks & contributions are welcome!

Contributors

Maintainer: mete0r

License

Disclosure

This program has been developed in accordance with a public document named “HWP Binary Specification 1.1” published by Hancom Inc.

CHAPTER 2

hwp5proc: HWPv5 processor

Do various operations on HWPv5 files.

Usage:

```
hwp5proc <command> [<args>...]
hwp5proc [--version]
hwp5proc [--help]
hwp5proc [--help-commands]

   --version        Show version and copyright information.
-h --help           Show help messages.
   --help-commands  Show available commands.
```

command: version

Print HWP file format version of <hwp5file>.

Usage:

```
hwp5proc version [options] <hwp5file>
hwp5proc version --help
```

Options:

```
-h --help              Show this screen
   --loglevel=<level>  Set log level.
   --logfile=<file>    Set log file.
```

command: header

Print HWP file header.

Usage:

```
hwp5proc header [options] <hwp5file>
hwp5proc header -h
```

Options:

```
-h --help              Show this screen
   --loglevel=<level>  Set log level.
   --logfile=<file>    Set log file.
```

command: `summaryinfo`

Print summary information of <hwp5file>.

Usage:

```
hwp5proc summaryinfo [options] <hwp5file>
hwp5proc summaryinfo --help
```

Options:

```
-h --help              Show this screen
   --loglevel=<level>  Set log level.
   --logfile=<file>    Set log file.
```

command: `ls`

List streams in the <hwp5file>.

Usage:

```
hwp5proc ls [--loglevel=<loglevel>] [--logfile=<logfile>]
            [--vstreams | --ole]
            <hwp5file>
hwp5proc ls --help
```

Options:

```
-h --help               Show this screen
   --loglevel=<level>   Set log level.
   --logfile=<file>     Set log file.

   --vstreams           Process with virtual streams (i.e. parsed/converted
                        form of real streams)
   --ole                Treat <hwpfile> as an OLE Compound File. As a
                        result, some streams will be presented as-is. (i.e.
                        not decompressed)
```

Example: List without virtual streams:

```
$ hwp5proc ls sample/sample-5017.hwp

\x05HwpSummaryInformation
BinData/BIN0002.jpg
BinData/BIN0002.png
BinData/BIN0003.png
BodyText/Section0
DocInfo
DocOptions/_LinkDoc
```

```
FileHeader
PrvImage
PrvText
Scripts/DefaultJScript
Scripts/JScriptVersion
```

Example: List virtual streams too:

```
$ hwp5proc ls --vstreams sample/sample-5017.hwp

\x05HwpSummaryInformation
\x05HwpSummaryInformation.txt
BinData/BIN0002.jpg
BinData/BIN0002.png
BinData/BIN0003.png
BodyText/Section0
BodyText/Section0.models
BodyText/Section0.records
BodyText/Section0.xml
BodyText.xml
DocInfo
DocInfo.models
DocInfo.records
DocInfo.xml
DocOptions/_LinkDoc
FileHeader
FileHeader.txt
PrvImage
PrvText
PrvText.utf8
Scripts/DefaultJScript
Scripts/JScriptVersion
```

command: `cat`

Extract out the specified stream in the <hwp5file> to the standard output.

Usage:

```
hwp5proc cat [--loglevel=<loglevel>] [--logfile=<logfile>]
             [--vstreams | --ole]
             <hwp5file> <stream>
hwp5proc cat --help
```

Options:

```
-h --help               Show this screen
   --loglevel=<level>   Set log level.
   --logfile=<file>     Set log file.

   --vstreams           Process with virtual streams (i.e. parsed/converted
                        form of real streams)
   --ole                Treat <hwpfile> as an OLE Compound File. As a
                        result, some streams will be presented as-is. (i.e.
                        not decompressed)
```

Example:

```
$ hwp5proc cat samples/sample-5017.hwp BinData/BIN0002.jpg | file -
```

```
$ hwp5proc cat samples/sample-5017.hwp BinData/BIN0002.jpg > BIN0002.jpg

$ hwp5proc cat samples/sample-5017.hwp PrvText | iconv -f utf-16le -t utf-8

$ hwp5proc cat --vstreams samples/sample-5017.hwp PrvText.utf8

$ hwp5proc cat --vstreams samples/sample-5017.hwp FileHeader.txt

ccl: 0
cert_drm: 0
cert_encrypted: 0
cert_signature_extra: 0
cert_signed: 0
compressed: 1
distributable: 0
drm: 0
history: 0
password: 0
script: 0
signature: HWP Document File
version: 5.0.1.7
xmltemplate_storage: 0
```

command: unpack

Extract out streams in the specified <hwp5file> to a directory.

Usage:

```
hwp5proc unpack [--loglevel=<loglevel>] [--logfile=<logfile>]
                [--vstreams | --ole]
                <hwp5file> [<out-directory>]
hwp5proc unpack --help
```

Options:

```
-h --help               Show this screen
   --loglevel=<level>   Set log level.
   --logfile=<file>     Set log file.

   --vstreams           Process with virtual streams (i.e. parsed/converted
                        form of real streams)
   --ole                Treat <hwpfile> as an OLE Compound File. As a
                        result, some streams will be presented as-is. (i.e.
                        not decompressed)
```

Example:

```
$ hwp5proc unpack samples/sample-5017.hwp
$ ls sample-5017
```

Example:

```
$ hwp5proc unpack --vstreams samples/sample-5017.hwp
$ cat sample-5017/PrvText.utf8
```

command: records

Print the record structure.

Usage:

```
hwp5proc records [--simple | --json | --raw | --raw-header | --raw-payload]
                 [--treegroup=<treegroup> | --range=<range>]
                 [--loglevel=<loglevel>] [--logfile=<logfile>]
                 <hwp5file> <record-stream>
hwp5proc records [--simple | --json | --raw | --raw-header | --raw-payload]
                 [--treegroup=<treegroup> | --range=<range>]
                 [--loglevel=<loglevel>] [--logfile=<logfile>]
hwp5proc records --help
```

Options:

```
-h --help               Show this screen
   --loglevel=<level>   Set log level.
   --logfile=<file>     Set log file.

   --simple             Print records as simple tree
   --json               Print records as json
   --raw                Print records as is
   --raw-header         Print record headers as is
   --raw-payload        Print record payloads as is

   --range=<range>      Print records specified in the <range>.
   --treegroup=<treegroup>
                        Print records specified in the <treegroup>.

<hwp5file>              HWPv5 files (*.hwp)
<record-stream>         Record-structured internal streams.
                        (e.g. DocInfo, BodyText/*)
<range>                 Specifies the range of the records.
                         N-M means "from the record N to M-1 (excluding M)"
                         N means just the record N
<treegroup>             Specifies the N-th subtree of the record structure.
```

Example:

```
$ hwp5proc records samples/sample-5017.hwp DocInfo
```

Example:

```
$ hwp5proc records samples/sample-5017.hwp DocInfo --range=0-2
```

If neither <hwp5file> nor <record-stream> is specified, the record stream is read from the standard input with an assumption that the input is in the format version specified by -V option.

Example:

```
$ hwp5proc records --raw samples/sample-5017.hwp DocInfo --range=0-2 > tmp.rec
$ hwp5proc records < tmp.rec
```

command: models

Print parsed binary models in the specified <record-stream>.

Usage:

```
hwp5proc models [--simple | --json | --format=<format> | --events]
                [--treegroup=<treegroup> | --seqno=<seqno>]
                [--loglevel=<loglevel>] [--logfile=<logfile>]
                (<hwp5file> <record-stream> | -V <version>)
hwp5proc models --help
```

Options:

```
-h --help               Show this screen
   --loglevel=<level>   Set log level.
   --logfile=<file>     Set log file.

   --simple             Print records as simple tree
   --json               Print records as json
   --format=<format>    Print records as formatted

   --treegroup=<treegroup>
                        Print records in the <treegroup>.
                        <treegroup> specifies the N-th subtree of the
                        record structure.
   --seqno=<seqno>      Print a model of <seqno>-th record

-V <version>, --file-format-version=<version>
                        Specifies HWPv5 file format version

<hwp5file>              HWPv5 files (*.hwp)
<record-stream>         Record-structured internal streams.
                        (e.g. DocInfo, BodyText/*)
```

Example:

```
$ hwp5proc models samples/sample-5017.hwp DocInfo
$ hwp5proc models samples/sample-5017.hwp BodyText/Section0

$ hwp5proc models samples/sample-5017.hwp docinfo
$ hwp5proc models samples/sample-5017.hwp bodytext/0
```

Example:

```
$ hwp5proc models --simple samples/sample-5017.hwp bodytext/0
$ hwp5proc models --format='%(level)s %(tagname)s\n' \
        samples/sample-5017.hwp bodytext/0
```

Example:

```
$ hwp5proc models --simple --treegroup=1 samples/sample-5017.hwp bodytext/0
$ hwp5proc models --simple --seqno=4 samples/sample-5017.hwp bodytext/0
```

If neither <hwp5file> nor <record-stream> is specified, the record stream is read from the standard input with an assumption that the input is in the format version specified by -V option.

Example:

```
$ hwp5proc cat samples/sample-5017.hwp BodyText/Section0 > Section0.bin
$ hwp5proc models -V 5.0.1.7 < Section0.bin
```

command: `find`

Find record models with specified predicates.

Usage:

```
hwp5proc find [--model=<model-name> | --tag=<hwptag>]
              [--incomplete] [--dump] [--format=<format>]
              [--loglevel=<loglevel>] [--logfile=<logfile>]
              (--from-stdin | <hwp5files>...)
hwp5proc find --help
```

Options:

```
-h --help                Show this screen
   --loglevel=<level>    Set log level.
   --logfile=<file>      Set log file.

   --from-stdin          get filenames fro stdin

   --model=<model-name> filter with record model name
   --tag=<hwptag>        filter with record HWPTAG
   --incomplete          filter with incompletely parsed content

   --format=<format>     record output format
                         %(filename)s %(stream)s %(seqno)s %(type)s
   --dump                dump record

<hwp5files>...           HWPv5 files (*.hwp)
```

Example: Find paragraphs:

```
$ hwp5proc find --model=Paragraph samples/*.hwp
$ hwp5proc find --tag=HWPTAG_PARA_TEXT samples/*.hwp
$ hwp5proc find --tag=66 samples/*.hwp
```

Example: Find and dump records of `HWPTAG_LIST_HEADER` which is parsed incompletely:

```
$ hwp5proc find --tag=HWPTAG_LIST_HEADER --incomplete --dump samples/*.hwp
```

command: `xml` (*Experimental*)

Transform an HWPv5 file into an XML.

Note: This command is experimental. Its output format is subject to change at any time.

Usage:

```
hwp5proc xml [--embedbin]
             [--no-xml-decl]
             [--output=<file>]
             [--format=<format>]
             [--no-validate-wellformed]
             [--loglevel=<loglevel>] [--logfile=<logfile>]
             <hwp5file>
hwp5proc xml --help
```

Options:

```
-h --help                Show this screen
   --loglevel=<level>    Set log level.
   --logfile=<file>      Set log file.

   --embedbin            Embed BinData/* streams in the output XML.
```

```
   --no-xml-decl         Don't output <?xml ... ?> XML declaration.
   --output=<file>       Output filename.

<hwp5file>               HWPv5 files (*.hwp)
<format>                 "flat", "nested" (default: "nested")
```

Example:

```
$ hwp5proc xml samples/sample-5017.hwp > sample-5017.xml
$ xmllint --format sample-5017.xml
```

With `--embedbin` option, you can embed base64-encoded `BinData/*` files in the output XML.

Example:

```
$ hwp5proc xml --embedbin samples/sample-5017.hwp > sample-5017.xml
$ xmllint --format sample-5017.xml
```

CHAPTER 3

Converters (*Experimental*)

Convert HWPv5 documents into other document formats.

Requirements

The conversions are performed with XSLT internally and verified with Relax NG if possible.

For these processing, the converters requires lxml (homepage) or libxml2's xsltproc / xmllint programs.

For lxml installation:

```
pip install --user lxml # install to user directory
pip install lxml        # install with virtualenv
```

or see Installing lxml.

(Currently conversions with lxml 2.3.5 is tested and verified to be working. lxml versions below that may work too, but those are not tested.)

For `xsltproc` / `xmllint` installation:

```
sudo apt-get install xsltproc libxml2-utils  # Debian/Ubuntu
```

Optional environment variables `PYHWP_XSLTPROC` and `PYHWP_XMLLINT` specifies the paths of the each programs. (If not set, `xsltproc` and/or `xmllint` should be in the one of the directories specified in `PATH`.)

`hwp5odt`: ODT conversion

HWPv5 to ODT converter

Usage:

```
hwp5odt [options] [--embed-image] <hwp5file>
hwp5odt [options] --styles <hwp5file>
hwp5odt [options] --content [--embed-image] <hwp5file>
hwp5odt [options] --document [--no-embed-image] <hwp5file>
hwp5odt -h | --help
hwp5odt --version
```

Options:

```
-h --help           Show this screen
--version           Show version
--loglevel=<level>  Set log level.
--logfile=<file>    Set log file.

--document          Produce single OpenDocument XML file (.fodt)
--styles            Produce *.styles.xml
--content           Produce *.content.xml

--output=<file>     Output file.
```

`hwp5html`: HTML conversion

HWPv5 to HTML converter

Usage:

```
hwp5html [options] <hwp5file>
hwp5html [options] <hwp5file> --html
hwp5html [options] <hwp5file> --css
hwp5html -h | --help
hwp5html --version
```

Options:

```
-h --help           Show this screen
--version           Show version
--loglevel=<level>  Set log level.
--logfile=<file>    Set log file.

--output=<output>   Output file / directory
```

`hwp5txt`: text conversion

HWPv5 to text converter

Usage:

```
hwp5txt [options] <hwp5file>
hwp5txt -h | --help
hwp5txt --version
```

Options:

```
-h --help           Show this screen
--version           Show version
--loglevel=<level>  Set log level.
--logfile=<file>    Set log file.

--output=<file>     Output file
```

CHAPTER 4

Hacking Guide

Standard procedures to hacking on `pyhwp`.

Contents:

Setup development environment

pyhwp project uses zc.buildout to manage the development environment. If you want to learn more about it, see buildout.

1. Install prerequisites

- CPython >= 2.6

  Although pyhwp itself can be working with CPython 2.5, various development helper scripts require CPython >= 2.6.

  In many GNU/Linux systems you can just install CPython with underlying package management system, e.g.

  ```
  sudo apt-get install python  # Debian/Ubuntu
  ```

  In MS-Windows systems, See Download Python.

- lxml

  In many GNU/Linux systems you can just install lxml with underlying package management system, e.g.

  ```
  sudo apt-get install python-lxml  # Debian/Ubuntu
  ```

  Or if your system has appropriate compilers, it will be installed automatically in later steps.

  In a MS-Windows system, you'll need install it manually. See Installing lxml.

  Note that this requirement will be removed in the future. See Issue #101.

- (optional) tox

  If you want to run full-blown tests, install tox.

2. Clone the source repository

```
$ git clone https://github.com/mete0r/pyhwp.git
```

3. Initialize the environment

Bootstrap buildout environment:

```
$ python bootstrap_me.py
$ python bootstrap.py  # bootstrap the buildout environment
```

Now there will be generated a **buildout** executable in the `bin/` directory.

Note: Bootstrapping the environment is required just only once for the first time.

Then run it to setup the environment:

```
$ bin/buildout
```

buildout will do following tasks:

- install development version of `pyhwp` scripts into the `bin/`
- generate configuration files for build/testing
- generate build/testing helper scripts
- ...

Note: Whenever the input configuration files (e.g. `buildout.cfg`, `tox.ini.in`, `setup.py`) get modified, you need to run **buildout** to update the environment again. However it's not required when the main source files get modified, i.e. the files under the `pyhwp/` directory and it's subdirectories.

Note: In this step, some optional components (e.g. JRE, multiple versions of Python installations) will be discovered and used by the relevant recipe and scripts.

4. Check basic stuffs

Run **hwp5proc**:

```
$ bin/hwp5proc --help
```

Do a quick test:

```
$ bin/test-core
```

Directory Layout

```
pyhwp                   Project Root
  |
  +-- pyhwp/            Source packages root
  |     |
  |     +-- hwp5/       Source package
```

```
|
+-- pyhwp-tests/      Test packages root
|     |
|     +-- hwp5_tests/ Test package
|
+-- docs/             Documentations, i.e. this document!
|
+-- bin/              hwp5proc, hwp5odt, build/testing scripts, etc.,
|
+-- etc/              development configuration files
|
+-- misc/             development configuration templates / helper scripts
|
+-- tools/            development helper packages
|
.
. (various directories)
.
```

After the initial *invocation of buildout* completes successfully, your directory will have a few more new generated directories, e.g. `bin/`, `develop-eggs/`. These are the standard buildout directories, which we will not cover the every details of them here. For general information, see Directory Structure of a Buildout.

Followings are `pyhwp` specific informations:

`/` - project root directory

The project root directory contains project configuration files.

`buildout.cfg` buildout configuration file.

`setup.py`, `setup.cfg` `pyhwp` setup files.

`tox.ini` tox configuration file. This file will be automatically generated from `tox.ini.in` by **`bin/buildout`**. See `[tox]` parts in `buildout.cfg`.

`tox.ini.in` tox configuration template file. If you want to modify tox configuration, edit this file and run **`bin/buildout`** again.

`bin/` - Buildout generated scripts

This directory will be populated with scripts generated from the `pyhwp` package and the various development helper packages/scripts.

`pyhwp` generate following scripts:

`hwp5proc` HWP format version 5 files processor. See *hwp5proc: HWPv5 processor*.

`hwp5odt`, `hwp5txt`, `hwp5html` Experimental converters. See *Converters (Experimental)*.

Development helper scripts (incomplete):

`buildout` (Re)generate the development environment.

`test-core` Run a quick unit test.

`pyhwp/` - the main source code

`hwp5/` The main source package. For now, there is not much documentation about the source code.

`pyhwp-tests/` - the main test suite

`hwp5_tests/` The main test suite.

`hwp5_xsl_tests/` XSLT test suite.

`hwp5_cli_tests.sh` Command-line interface tests.

`tools/` - Development helper packages

`discover.python/` `discover.lxml/` `discover.jre/` `discover.lo/` `install.jython/`

Discover multiple python versions, lxml, JRE, Libreoffice to use in the developement environment. Provides zc.buildout recipes.

`xsltest/`

an XSLT test runner.

`oxt.tool/`

Build and test .oxt packages with the LibreOffice.

Hack & Test

If you modify some modules in `hwp5` package in the `pyhwp/` directory, you can test the modification with the `hwp5proc` script in the `bin/` directory.

You can test the `hwp5` package by executing `bin/test-core`, but it's just a quick test and not a complete test suite. If you want to run a full-blown test suite, run `tox`, which tries to test `pyhwp` in various virtualenv-isolated python platforms, including Python 2.5, 2.6, 2.7, Jython 2.5 and PyPy.

```
$ bin/buildout

(...)

$ vim pyhwp/hwp5/proc/__init__.py

(HACK HACK HACK)

$ bin/test-core

$ bin/hwp5proc ...

$ bin/tox
```

CHAPTER 5

CHANGES

0.1b9 (unreleased)

- Nothing changed yet.

0.1b8 (2014-11-03)

- hwp5view: experimental viewer with webkitgtk+
- hwp5proc: xml –formats (“flat”, “nested”)
- hwp5proc: models –events (experimental)
- hwp5proc: models –seqno –format (incompatible changes)
- hwp5proc: find –from-stdin
- hwp5proc: find –format
- binmodels: GShapeObjectCaption
- olestorage: Gsf implementation through python-gi
- olestorage: use new olefile instead of OleFileIO_PL

0.1b7 (2014-01-31)

- support distribution docs. (based on Changwoo Ryu’s algorithm)

0.1b6 (2014-01-20)

- binmodel: change type of TableCell dimensions to signed integer
- hwp5odt: fix NCName for style:name (close #140)
- hwp5proc: fix with-statement in ‘xml’ command for Python 2.5

- hwp5proc: mark 'xml' command experimental

0.1b5 (2013-10-29)

- close #134
- hwp5html generates .xhtml instead of .html
- hwp5proc: new '–no-xml-decl' option
- hwp5odt: fix to not use '/' in resulting style names
- hwp5proc: IdMappings.memoshape only if version > 5.0.1.6

0.1b4 (2013-07-03)

- hwp5proc records: new option '–raw-header'
- hwp5odt: new '–document' option produces single ODT XML files (`*.fodt`)
- hwp5odt: new '–styles', '–content' option produces styles/content XML files
- ODT XSL files restructured

0.1b3 (2013-06-18)

- Fix IdMappings (#125)
- hwp5proc records: new option '–raw-payload'
- hwp5proc xml: FlagsType as xsd:hexBinary
- Various binary/xml models changes

0.1b2 (2013-06-08)

- Add PyPy support

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

H