CheckPPML Pro 3.0 and CheckPPML 3.0

CheckPPML Pro and the original CheckPPML can be used to view and validate PPML files. The original CheckPPML is limited to processing up to 100 pages per file and can be downloaded for free. The Pro version allows for an unlimited number of pages to be processed.

CheckPPML is a PPML Consumer that supports the GA (Graphics Arts) subset of PPML versions 1.5, 2.1, 2.2 and 3.0. It supports relative and absolute URL's as well as URL's using the http, https, ftp and file protocols.

CheckPPML has both a command line and a simple graphical user interface to convert PPML files to PDF.

Installing CheckPPML on a PC

You will need the following to be able to run CheckPPML:

- Java 1.5 Run Time Environment (JRE) http://java.com -or- Development Kit (JDK), downloadable from http://java.sun.com
- GPL GhostScript -or- Adobe Distiller (a component of Acrobat commercial versions, but not Acrobat Reader). GhostScript for Windows PC can be downloaded from http://www.softpedia.com/progDownload/GPL-Ghostscript-Download-83803.html

After making sure that the above tools are installed, you may unpack CheckPPML zip package into any directory on your system. You should be able to double-click on CheckPPML.jar to start the graphical user interface.

Installing CheckPPML on a Mac

CheckPPML runs well on Mac OSX 10.4.9 and later using either the built-in PostScript normalizer or GPL Ghostscript. To use GhostScript the gs executable must be in the users PATH (which is the case when you install Ghostscript in it's default location on the Mac). Like the Windows version you may install CheckPPML anywhere and can double click on the CheckPPML.jar file to launch the GUI.

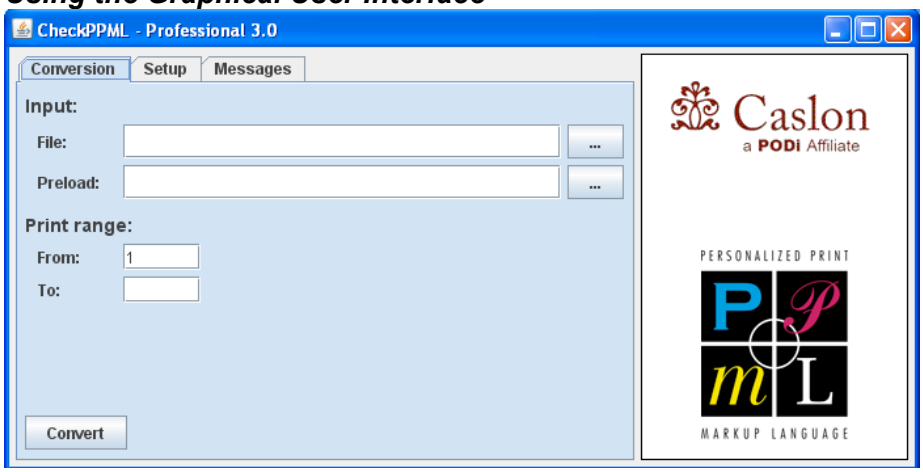
- GPL Ghostscript for Mac can be downloaded from http://www.openprinting.org/download/printdriver/macosx/
  The correct file is named gplgs-8.61xxx.dmg or higher.

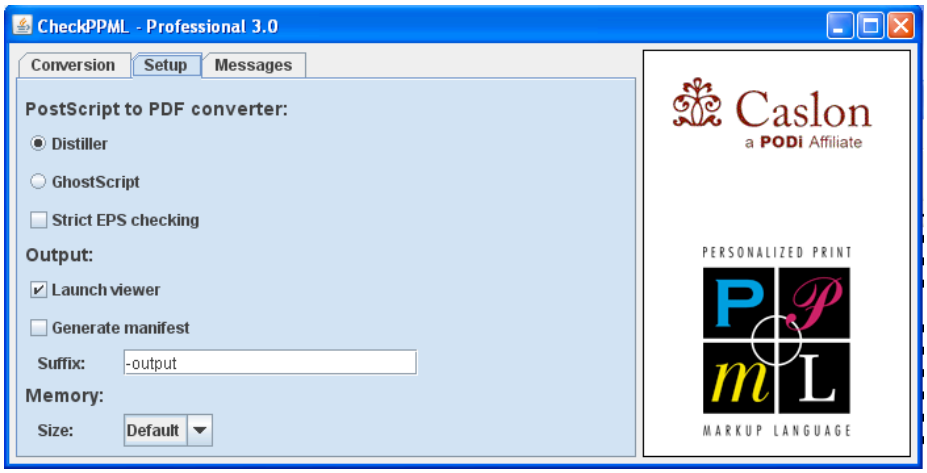
CheckPPML also runs well on Intel Macs that run Microsoft Windows in a Parallels Desktop (www.parallels.com) or VMWare Fusion (www.vmware.com) virtual PC window. Please note that due to differences in libraries used on the Mac platform not all image formats are as well supported as on the Windows platform.

Transparency support

Please note that Ghostscript does not properly render transparency between layers. Transparency is only accurately represented in the results when using Acrobat Professional on a Windows PC.

Using the Graphical User Interface





The Graphical User Interface (GUI) has three tabs: Conversion, Setup and Messages:

- On the "Conversion" tab, clicking on the button next to "File:" will allow you to select the PPML file you would like to check and convert to PDF. If this PPML file depends on global reusable object definitions, then click on the button next to "Preload:" to select the PPML file that defines those global reusable object definitions. Please note that no output will be generated for this PPML preload file.
- Under the "Setup tab, in the PS to PDF converter section you can choose to use GhostScript or Adobe Distiller (or Built-in on Mac OSX). The selected distiller is used whenever CheckPPML needs to convert PostScript to PDF. Please note that CheckPPML only uses this converter when it encounters PostScript files, CheckPPML does not need it for PPML files that do not refer to PostScript files. The “Strict EPS Checking” checkbox selects whether CheckPPML will perform EPS conformance checks on the PostScript content encountered by CheckPPML. Some PPML consumers are able to successfully process files with these EPS errors. The “Convert ProcSet to Prolog” implements a work around for PPML Producers that incorrectly assume that a PostScript ProcSet acts like a PostScript prolog. Some PPML consumers are not as strict as CheckPPML enabling such files to print successfully anyway. The "Print range" section allows you to select a range of pages for which CheckPPML should generate PDF output. The “Suffix” field is used to determine the names of the various output files generated by CheckPPML. The “Memory” field enables large PPML files to be processed by increasing the amount of memory available to CheckPPML.

- The “Messages” tab will show messages that are generated during the conversion. The generated log file contains the same information.

Output file naming

The names of the output files are all placed in the same directory as the main PPML file and have a name derived from the name of the main PPML file. If the main PPML file is called example.ppml then:

- The generated PDF file will be called example<suffix>.pdf
- The manifest file will be called example<suffix>-manifest.xml
- The log file (containing error and warning messages) will be called example<suffix>.log

Memory Usage

Images consume huge amounts of memory, and it is definitely possible to run out. CheckPPML default is limited to Java's 64MB of virtual memory by default. This may not be enough, but there is a solution. Under the "Setup" tab, up to 2GB of memory may be allocated to CheckPPML.

Memory required is approximately 220% of the largest single image, when un-packed. As an example, a 32MB JPEG occupies 5 to 10X this space in memory. So, if we presume 10X, a 32MB JPEG becomes 320MB, and CheckPPML will require 220% larger memory, or 700MB of RAM to run (in addition to memory required for basic system functions).

If on the Messages tab an "out of memory" message is displayed when checking a file, consider changing the amount of memory allocated by CheckPPML on the Setup tab.

CheckPPML's Manifest feature

CheckPPML's default setting is to generate a Manifest listing (XML) of all components of the PPML dataset during the checking process. The manifest is placed in the same location as the PPML dataset, and contains a unique MD5 checksum value for the content of the specific PPML dataset, and MD5 checksums for all the assets referred to by that PPML file. The recipient of a PPML dataset and manifest can use CheckPPML to verify that all assets required for the job have been received by re-running CheckPPML on the package as received with a different output suffix. If the checksums in the newly generated manifest matches the manifest in the received package, nothing has changed, and the PDFs generated by CheckPPML at point of origin will also match the one generated upon receipt.

Using the command line interface
CheckPPML can also be run from the command line. It must be run as:
java -jar <*installation-directory*>/manifest.jar <*parameters*>

The parameters are as follows:

- **-o <*output file*>**: instructs CheckPPML to generate a manifest file with the given name
- **-pdf <*pdf file*>**: instructs CheckPPML to convert the PPML file to a PDF file with the given name
- **-distiller**: instructs CheckPPML to use Adobe Distiller for PostScript to PDF conversion
- **-gs**: instructs CheckPPML to use AFPL GhostScript for PostScript to PDF conversion
- **-checkEPS**: instruct CheckPPML to perform EPS conformance tests on PostScript content
- **-debug**: instruct CheckPPML to keep the generated PostScript files for debugging purposes
- **-fixprolog**: instruct CheckPPML to convert a PostScript ProcSet to a Prolog.
- **-from <*pagenr*>**: instruct CheckPPML to only create output for pages starting at the specified page number
- **-to <*pagenr*>**: instruct CheckPPML to only create output for pages up to the specified page number
- **-log <*log file*>**: instruct CheckPPML to generate error and warning messages to a log file instead of generating messages on the standard error output
- **-s <*directory*>**: instructs CheckPPML to lookup relative file names in the PPML file in the given directory instead of the directory in which the PPML file is located
- **-d <*description*>**: defines the descriptive text to be used in the manifest file
- **-f <*feature name*> <*category name*>**: defines the name of a feature and category to be included in the manifest file (may be specified multiple times)
- the remaining arguments are assumed to be PPML files that must be processed. Output will only be generated for the last PPML file; the other PPML files are processed to gather global reusable object definitions.

Reporting issues
Please report issues to ppmlinfo@podi.org. Please include:

- a description of the problem
- the PPML file and any files referenced from that PPML file
- a PDF file showing the correct output

Requesting features
Feature request may be sent to ppmlinfo@podi.org. Please include:

- a short description of the feature
- a business case for adding the feature

Change Log 3.0.6
The following enhancements have been made in CheckPPML-3.0.6:
- Improved detection of installed Distiller versions

Change Log 3.0.3
The following enhancements have been made in CheckPPML-3.0.3:
- Improved detection of installed Distiller versions
- Include a patch to allow a ProcSet to be converted into a Prolog

The following issues have been fixed in CheckPPML-3.0.3:
- Change default softmask background to be white instead of black

Change Log 3.0
The following enhancements have been made in CheckPPML-3.0:
- improved performance
- support for PPML/GA 3 datasets including transparency

Change Log 1.2
The following enhancements have been made in CheckPPML-1.2:
- improved performance
- added support for allocating more memory to CheckPPML
- improved user interface and error handling

Change Log 1.1
The following enhancements have been made in CheckPPML-1.1:
- improved performance
- added support for Adobe CMYK JPEG images
- added support for the Mac OSX built-in PostScript Normalizer

The following issues have been fixed in CheckPPML-1.1:
- fixed content errors when processing ZIP files
- fixed content errors when processing large numbers of REUSABLE_OBJECT definitions

Change Log 1.0.5
The following issues have been fixed in CheckPPML-1.0.5:
- fixed detection of content size for TIFF and JPEG images
- fixed detection of Acrobat Distiller 8

Change Log 1.0.4
The following enhancements have been made in CheckPPML-1.0.4:
- improved performance
- reduced memory usage

Change Log 1.0.3b1
The following issues have been fixed in CheckPPML-1.0.3b1:
- fixed inheritance of PAGE_DESIGN elements
- fixed problems with the usage of Acrobat Distiller 8.1

The following enhancements have been made in CheckPPML-1.0.3b1:
- added support for Prolog style supplied resources
- enhanced verification of supplied resource content
- added EPS conformance checking for PostScript content
- added PODi and PPML logos

Change Log 1.0.2
The following issues have been fixed in CheckPPML-1.0.2:
- erroneous complaints about an Overwrite attribute on SUPPLIED_RESOURCE and OCCURRENCE elements (schema issue)
- fixed problems with empty PS files
- fixed problems with multi-page TIFF files (incorrect clipping for certain tiled TIFF's)

The following enhancements have been made:

- reduced memory usage when dealing with raster image data.
- checks were added to detect when the Dimensions attribute of a SOURCE element does not match the dimensions of the content referred to from that SOURCE element.

Change Log 1.0.1

The following issues have been fixed in CheckPPML-1.0.1:

- after switching from Distiller to GhostScript CheckPPML uses the incorrect path for the PostScript interpreter
- the checkppml.joboptions file would not be found if the path to the CheckPPML installation contained spaces
- CheckPPML hangs if the conversion runs out of memory (which is limited by the Java VM at startup)
- Base64 decoding sometimes adds an extra byte to the decoded content
- Certain attributes do not allow scientific notation or numbers starting with a decimal sign (.5) to be used
- TIFF files without an Xresolution/YResolution unit cannot be processed
- Redefining global reusable objects produces errors
- Defining global supplied resources produces errors
- Deleting global reusable objects or supplied resources produces errors
- Processing a file with a CONFORMANCE element produces errors
- The path to Distiller and/or GhostScript is not correctly detected
- PDF output of CheckPPML produces a “Drawing error” (issue in Acrobat please use version 8)
- PPML files processed by CheckPPML can not be modified while CheckPPML is still running
- EPS files with a binary preview header cause conversion errors

Furthermore CheckPPML-1.0.1 now supports ZIPCD packaged PPML datasets.