GNU Emacs package: rcd-paps.el

By using the package RCD Paps for GNU Emacs user may generate PDF, PS or SVG files for current buffer interactively or by using program. This is very useful to print Emacs buffers and files.

Where to find RCD Paps for Emacs

Package source:
https://gnu.support/files/emacs/packages/rcd-paps.el

GNU Emacs package: rcd-paps.el:
https://gnu.support/gnu-emacs/packages/GNU-Emacs-package-rcd-paps-el-76862.html

and PDF version of this page:
https://gnu.support/files/emacs/packages/rcd-paps/GNU-Emacs-package-rcd-paps-el-76862.pdf

Installation instruction

RCD Dashboard package is compatible with Emacs 29 version.

Install the following package required for RCD Org Agenda Dashboard:

GNU Emacs package: rcd-utilities.el:
https://gnu.support/gnu-emacs/packages/rcd-utilities-el.html

and of course you have to install `paps` command line utility. □□ It may be part of your GNU/Linux Operating System and so try using your package manager to find the `paps` or download it directly from sources.

`paps` can print Emoji

The `paps` utility can print Emacs emoji and buffers of all kinds.

Sample PDF output: https://gnu.support/files/emacs/packages/rcd-paps/2022-12-30-17:04:00-1.pdf

Sample PDF output is here, GNU/Emacs Tutorial printed with Paps: https://gnu.support/files/emacs/packages/rcd-paps/2023-01-01-02:42:51.pdf

1. Download the package from: https://gnu.support/files/emacs/packages/rcd-paps.el
2. Install the package by using:

```
{M-x package-install-file RET rcd-paps.el RET}
```

Usage instructions for RCD Paps for Emacs

1. Setup RCD Paps for Emacs by using `M-x customize-group RET rcd-paps RET`
2. Command `M-x rcd-paps-buffer-to-pdf` is to generate PDF file for current buffer as PDF. Corresponding `rcd-paps-buffer-to-pdf-view` command will open the PDF in PDF reader.
3. Command `M-x rcd-paps-region-to-pdf` is to generate PDF file for selected region as PDF. Corresponding `rcd-paps-region-to-pdf-view` command will open the PDF.
4. To setup header or footer, use `M-x rcd-paps-setup-header` or `M-x rcd-paps-setup-footer` options. Use general `rcd-paps-setup-header-and-footer` option.
5. There are many improvements to be done for this package, send your suggestions to author.

Source of rcd-paps.el

```
;;; rcd-paps.el --- RCD PAPS Emacs Bindings to PAPS printing utilities  -*- lexical-
binding: t; -*-

;; Copyright (C) 2021-2022 by Jean Louis

;; Author: Jean Louis <bugs@gnu.support>
;; Version: 0.2
;; Package-Requires: (rcd-utilities)
;; Keywords: tools wp multimedia processes
;; URL: https://gnu.support/gnu-emacs/packages/GNU-Emacs-package-rcd-paps-el-
76862.html

;; This file is not part of GNU Emacs.

;; This program is free software: you can redistribute it and/or
;; modify it under the terms of the GNU General Public License as
;; published by the Free Software Foundation, either version 3 of the
;; License, or (at your option) any later version.
;;
;; This program is distributed in the hope that it will be useful, but
;; WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
;; General Public License for more details.
;;
;; You should have received a copy of the GNU General Public License
;; along with this program. If not, see <http://www.gnu.org/licenses/>.

;;; Commentary:

;;; Change Log:
```

```
;;; Code:

(require 'rcd-utilities)

(defcustom rcd-paps-output-directory (file-name-as-directory (getenv "HOME"))
  "Default PAPS output directory."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-pdf-viewer "evince"
  "PDF Viewer program that must be found in executable path."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-orientation nil
  "Page orientation. By default it is portrait, if TRUE, it will
be landscape."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-columns 1
  "Number of columns output."
  :group 'rcd-paps
  :type 'integer)

(defcustom rcd-paps-font "Monospace 12"
  "Set font for `paps'."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-rtl nil
  "When TRUE, do right-to-left text layout."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-separation-lines nil
  "When TRUE, draw separation lines between columns and headers and footers."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-justify nil
  "When TRUE, justify the layout."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-hyphens nil
  "When TRUE, show hyphens when wrapping."
  :group 'rcd-paps
  :type 'boolean)
```

```
 (defcustom rcd-paps-show-wrap nil
   "When TRUE, show characters for wrapping."
   :group 'rcd-paps
   :type ''(choice
       (string :tag "Default wrap by word-char" :value "word-char")
       (string :tag "Wrap by word" :value "word")
       (string :tag "Wrap by char" :value "char")))

  (defcustom rcd-paps-paper-size "a4"
   "Set paper size [legal, letter, a3, a4]. (Default: a4)."
   :group 'rcd-paps
   :type '(choice
       (string :tag "Paper size A4" :value "a4")
       (string :tag "Paper size A3" :value "a3")
       (string :tag "Paper size letter" :value "letter")
       (string :tag "Paper size legal" :value "legal")))

  (defcustom rcd-paps-glyph-gravity "auto"
   "Set base glyph rotation (\"south\" \"west\" \"north\" \"east\"
\"auto\"), default is \"auto\"."
   :group 'rcd-paps
   :type '(choice
       (string :tag "Default glyph rotation auto" :value "auto")
       (string :tag "Glyph rotation south" :value "south")
       (string :tag "Glyph rotation west" :value "west")
       (string :tag "Glyph rotation north" :value "north")
       (string :tag "Glyph rotation east" :value "east")))

  (defcustom rcd-paps-glyph-gravity-hint "natural"
   "Set base glyph orientation (\"natural\" \"strong\" \"line\"),
default is \"natural\"."
   :group 'rcd-paps
   :type '(choice
       (string :tag "Default glyph orientation natural" :value "natural")
       (string :tag "Glyph orientation strong" :value "strong")
       (string :tag "Glyph orientation line" :value "line")))

  (defcustom rcd-paps-output-format "pdf"
   "Set output format (\"pdf\", \"svg\" or \"ps), default is \"pdf\"."
   :group 'rcd-paps
   :type '(choice
       (string :tag "Default output format PDF" :value "pdf")
       (string :tag "Output format Postscript" :value "ps")
       (string :tag "Output format SVG" :value "svg")))

  (defcustom rcd-paps-bottom-margin 36
   "Set bottom margin in postscript point units (1/72 inch). (Default: 36)"
   :group 'rcd-paps
   :type 'integer)

  (defcustom rcd-paps-top-margin 36
```

```
  "Set top margin in postscript point units (1/72 inch). (Default: 36)"
  :group 'rcd-paps
  :type 'integer)

 (defcustom rcd-paps-right-margin 36
  "Set right margin in postscript point units (1/72 inch). (Default: 36)"
  :group 'rcd-paps
  :type 'integer)

 (defcustom rcd-paps-left-margin 36
  "Set left margin in postscript point units (1/72 inch). (Default: 36)"
  :group 'rcd-paps
  :type 'integer)

(defcustom rcd-paps-gutter-width 40
  "Set gutter widthd. (Default: 40)"
  :group 'rcd-paps
  :type 'integer)

(defcustom rcd-paps-print-header t
  "When TRUE, print page header."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-print-footer t
  "When TRUE, print page footer."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-title ""
  "Title string for page header (Default: filename/stdin)."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-header-left ""
  "Left side of the header. Default is localized date."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-header-center ""
  "Center side of the header. Default is localized date."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-header-right ""
  "Right side of the header. Default is localized date."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-footer-left ""
  "Left side of the footer. Default is localized date."
```

```
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-footer-center ""
  "Center side of the footer. Default is localized date."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-footer-right ""
  "Right side of the footer. Default is localized date."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-pango nil
  "When TRUE, interpret input text as pango markup."
  :group 'rcd-paps
  :type 'boolean)

(defcustom rcd-paps-encoding "UTF-8"
  "Assume encoding of input text, default is \"UTF-8\"."
  :group 'rcd-paps
  :type 'string)

(defcustom rcd-paps-lines-per-inch 0
  "Set the amount of lines per inch. Zero `0' means defaults will be used."
  :group 'rcd-paps
  :type 'integer)

(defcustom rcd-paps-characters-per-inch 0
  "Set the amount of characters per inch. Zero `0' means defaults will be used."
  :group 'rcd-paps
  :type 'integer)

(defvar-local rcd-paps-arguments nil
  "RCD Paps arguments, variable local to used buffer.")

(defun rcd-paps-setup-arguments ()
  "Return list to be used as arguments ARGS for function `rcd-paps-process'."
  (setq-local rcd-paps-arguments nil)
  (setq-local rcd-paps-arguments (cons (format "--format=%s" rcd-paps-output-format)
rcd-paps-arguments))
  (when rcd-paps-orientation
    (setq rcd-paps-arguments (cons "--landscape" rcd-paps-arguments)))
  (when rcd-paps-columns
    (setq rcd-paps-arguments (cons (format "--columns=%d" rcd-paps-columns) rcd-paps-
arguments)))
  (when rcd-paps-justify
    (setq rcd-paps-arguments (cons "--justify" rcd-paps-arguments)))
  (when rcd-paps-hyphens
    (setq rcd-paps-arguments (cons "--hyphens" rcd-paps-arguments)))
  (when rcd-paps-pango
```

```
    (setq rcd-paps-arguments (cons "--markup" rcd-paps-arguments)))
  (when rcd-paps-rtl
    (setq rcd-paps-arguments (cons "--rtl" rcd-paps-arguments)))
  (when rcd-paps-separation-lines
    (setq rcd-paps-arguments (cons "--separation-lines" rcd-paps-arguments)))
  (when rcd-paps-show-wrap
    (setq rcd-paps-arguments (cons (format "--wrap=%s" rcd-paps-show-wrap) rcd-paps-
arguments)))
  (when rcd-paps-font
    (setq rcd-paps-arguments (cons (format "--font=%s" rcd-paps-font) rcd-paps-
arguments)))
  (when rcd-paps-paper-size
    (setq rcd-paps-arguments (cons (format "--paper=%s" rcd-paps-paper-size) rcd-paps-
arguments)))
  (unless (string-empty-p rcd-paps-title)
    (setq rcd-paps-arguments (cons (format "--title=%s" rcd-paps-title) rcd-paps-
arguments)))
  (when rcd-paps-print-header
    (setq rcd-paps-arguments (cons "--header" rcd-paps-arguments)))
  (when rcd-paps-print-footer
    (setq rcd-paps-arguments (cons "--footer" rcd-paps-arguments)))
  (when rcd-paps-glyph-gravity
    (setq rcd-paps-arguments (cons (format "--gravity=%s" rcd-paps-glyph-gravity) rcd-
paps-arguments)))
  (when rcd-paps-glyph-gravity-hint
    (setq rcd-paps-arguments (cons (format "--gravity-hint=%s" rcd-paps-glyph-gravity-
hint) rcd-paps-arguments)))
  (when rcd-paps-bottom-margin
    (setq rcd-paps-arguments (cons (format "--bottom-margin=%d" rcd-paps-bottom-
margin) rcd-paps-arguments)))
  (when rcd-paps-top-margin
    (setq rcd-paps-arguments (cons (format "--top-margin=%d" rcd-paps-top-margin) rcd-
paps-arguments)))
  (when rcd-paps-right-margin
    (setq rcd-paps-arguments (cons (format "--right-margin=%d" rcd-paps-right-margin)
rcd-paps-arguments)))
  (when rcd-paps-left-margin
    (setq rcd-paps-arguments (cons (format "--left-margin=%d" rcd-paps-left-margin)
rcd-paps-arguments)))
  (when rcd-paps-gutter-width
    (setq rcd-paps-arguments (cons (format "--gutter-width=%d" rcd-paps-gutter-width)
rcd-paps-arguments)))
  (unless (string= rcd-paps-encoding "UTF-8")
    (setq rcd-paps-arguments (cons (format "--encoding=%s" rcd-paps-encoding) rcd-
paps-arguments)))
  (when (and rcd-paps-characters-per-inch (not (= rcd-paps-characters-per-inch 0)))
    (setq rcd-paps-arguments (cons (format "--cpi=%d" rcd-paps-characters-per-inch)
rcd-paps-arguments)))
  (unless (and rcd-paps-lines-per-inch (= rcd-paps-lines-per-inch 0))
    (setq rcd-paps-arguments (cons (format "--lpi=%d" rcd-paps-lines-per-inch) rcd-
paps-arguments)))
```

```
  (unless (string-empty-p rcd-paps-header-left)
    (setq rcd-paps-arguments (cons (format "--header-left=%s" rcd-paps-header-left)
rcd-paps-arguments)))
  (unless (string-empty-p rcd-paps-header-right)
    (setq rcd-paps-arguments (cons (format "--header-right=%s" rcd-paps-header-right)
rcd-paps-arguments)))
  (unless (string-empty-p rcd-paps-header-center)
    (setq rcd-paps-arguments (cons (format "--header-center=%s" rcd-paps-header-
center) rcd-paps-arguments)))
  (unless (string-empty-p rcd-paps-footer-left)
    (setq rcd-paps-arguments (cons (format "--footer-left=%s" rcd-paps-footer-left)
rcd-paps-arguments)))
  (unless (string-empty-p rcd-paps-footer-right)
    (setq rcd-paps-arguments (cons (format "--footer-right=%s" rcd-paps-footer-right)
rcd-paps-arguments)))
  (unless (string-empty-p rcd-paps-footer-center)
    (setq rcd-paps-arguments (cons (format "--footer-center=%s" rcd-paps-footer-
center) rcd-paps-arguments)))
  rcd-paps-arguments)

(defun rcd-paps-setup-header-footer-help ()
  "Show help for footer and header settings."
  (rcd-pop-to-report "
Options for footer and header take a string that is based on
python f-strings. Following variable may be used in the
definitions of the headers and footer:

   path - The complete path of the file being printed
   filename - The basename (without directory) of the file being printed
   mtime - The file's modification time
   now - The current time
   num_pages - The total number of pages in the document
   page_idx - The current page being printed

Here is an example of how it may be used:

paps --header-left=\"{now:%Y-%m-%d %H:%M}\" --header-center=\"{filename}\" --header
-right=\"Page {page_idx:02d}/{num_pages:02d}\" --header -o hello.pdf paps.cc"
             "*RCD Paps Help*"))

(defun rcd-paps-setup-header ()
  "Interactively setup header for `paps'."
  (interactive)
  (let ((current-buffer (current-buffer)))
    (rcd-paps-setup-header-footer-help)
    (switch-to-buffer current-buffer)
  (setq rcd-paps-print-header (y-or-n-p "Print headers?"))
  (when rcd-paps-print-header
    (let ((left-header (rcd-ask "Left header: "))
      (center-header (rcd-ask "Header center: "))
      (right-header (rcd-ask "Right header: ")))
```

```
      (setq rcd-paps-header-left left-header)
      (setq rcd-paps-header-right right-header)
      (setq rcd-paps-header-center center-header)
      (kill-buffer "*RCD Paps Help*")))))

(defun rcd-paps-setup-footer ()
  "Interactively setup footer for `paps'."
  (interactive)
  (let ((current-buffer (current-buffer)))
    (rcd-paps-setup-header-footer-help)
    (switch-to-buffer current-buffer)
  (setq rcd-paps-print-footer (y-or-n-p "Print footer?"))
  (when rcd-paps-print-footer
    (let ((left-footer (rcd-ask "Left footer: "))
      (center-footer (rcd-ask "Footer center: "))
      (right-footer (rcd-ask "Right footer: ")))
      (setq rcd-paps-footer-left left-footer)
      (setq rcd-paps-footer-right right-footer)
      (setq rcd-paps-footer-center center-footer)))
  (kill-buffer "*RCD Paps Help*")))

(defun rcd-paps-setup-header-and-footer ()
  "Interactively setup header and footer."
  (interactive)
  (rcd-paps-setup-header-footer-help)
  (rcd-paps-setup-header)
  (rcd-paps-setup-footer))

(defun rcd-paps-process (text &rest args)
  "Return `paps' processed output by using TEXT.

ARGS is optional list of arguments."
  (let* ((args (or (delq nil args) (rcd-paps-setup-arguments))))
    (cond (args (apply 'rcd-command-output-from-input "paps" text args))
      (t (rcd-command-output-from-input "paps" text)))))

(defun rcd-paps-output-file ()
  "Return the output file for `paps'."
  (concat (file-name-as-directory rcd-paps-output-directory) (rcd-timestamp) ".pdf"))

(defun rcd-paps-read-file ()
  "Return output file for paps. With `C-u' prefix, ask for file name."
  (cond (current-prefix-arg (read-file-name
                "File name for output: "
                rcd-paps-output-directory
                (concat (buffer-name) rcd-paps-output-format)))
    (t (rcd-paps-output-file))))

(defun rcd-paps-process-buffer (&rest args)
  "Return current buffer processed by command `paps'.
```

```
ARGS is optional list of arguments."
  (let ((text (buffer-substring-no-properties (point-min) (point-max))))
    (apply 'rcd-paps-process text args)))

(defun rcd-paps-buffer-to-pdf (&optional file-name &rest args)
  "Generate PDF for current buffer by using `paps' command."
  (interactive)
  (let ((output (rcd-paps-process-buffer args))
    (file-name (or file-name (rcd-paps-read-file))))
    (prog1
    (string-to-file-force output file-name)
    (rcd-message "PDF: %s" file-name))))

(defun rcd-paps-buffer-to-pdf-view (&rest args)
  "Process current buffer with `paps' and open PDF."
  (interactive)
  (start-process rcd-paps-pdf-viewer rcd-paps-pdf-viewer rcd-paps-pdf-viewer (rcd-
paps-buffer-to-pdf args)))

(defun rcd-paps-text-to-pdf (text &optional file-name &rest args)
  "Use command `paps' to generate PDF file for TEXT.

FILE-NAME is optional.
ARGS are optional arguments. When ARGS is used, no global arguments are applied."
  (let* ((output (apply 'rcd-paps-process text args))
     (file-name (or file-name (rcd-paps-read-file))))
    (string-to-file-force output file-name)))

(defun rcd-paps-process-region (&rest args)
  "Process region with `paps'."
  (let ((text (rcd-region-string)))
    (when text
      (apply 'rcd-paps-process text args))))

(defun rcd-paps-region-to-pdf (&optional args)
  "Process current region with `paps' and save to PDF."
  (interactive)
  (let ((output (rcd-paps-process-region args))
    (file-name (rcd-paps-read-file)))
    (prog1
    (string-to-file-force output file-name)
    (rcd-message "PDF: %s" file-name))))

(defun rcd-paps-region-to-pdf-view (&optional args)
  "Process current buffer with `paps' and open PDF."
  (interactive)
  (start-process rcd-paps-pdf-viewer rcd-paps-pdf-viewer rcd-paps-pdf-viewer (rcd-
paps-region-to-pdf args)))

(provide 'rcd-paps)
```

```
;;; rcd-paps.el ends here
```

Source of this page

```
:icons: font
:source-highlighter: rouge

By using the package RCD Paps for GNU Emacs user may generate PDF, PS
or SVG files for current buffer interactively or by using
program. This is very useful to print Emacs buffers and files.

== Where to find RCD Paps for Emacs

Package source: +
https://gnu.support/files/emacs/packages/rcd-paps.el

GNU Emacs package: rcd-paps.el: +
https://gnu.support/gnu-emacs/packages/GNU-Emacs-package-rcd-paps-el-76862.html

and PDF version of this page: +
https://gnu.support/files/emacs/packages/rcd-paps/GNU-Emacs-package-rcd-paps-el-
76862.pdf


== Installation instruction

CAUTION: RCD Dashboard package is compatible with Emacs 29 version.

Install the following package required for RCD Org Agenda Dashboard:

GNU Emacs package: rcd-utilities.el: +
https://gnu.support/gnu-emacs/packages/rcd-utilities-el.html +

and of course you have to install `paps` command line utility. □□ It
may be part of your
https://www.gnu.org/distros/free-distros.html[GNU/Linux Operating
System] and so try using your package manager to find the `paps` or
https://github.com/dov/paps[download it directly from sources].

== `paps` can print Emoji

The `paps` utility can print Emacs emoji and buffers of all kinds.

NOTE: Sample PDF output:
https://gnu.support/files/emacs/packages/rcd-paps/2022-12-30-17:04:00-1.pdf

NOTE: Sample PDF output is here, GNU/Emacs Tutorial printed with Paps:
https://gnu.support/files/emacs/packages/rcd-paps/2023-01-01-02:42:51.pdf
```

````
1. Download the package from: <https://gnu.support/files/emacs/packages/rcd-paps.el>

2. Install the package by using: +
```
{M-x package-install-file RET rcd-paps.el RET}
```

== Usage instructions for RCD Paps for Emacs

1. Setup RCD Paps for Emacs by using `M-x customize-group RET rcd-paps RET`

2. Command `M-x rcd-paps-buffer-to-pdf` is to generate PDF file for
current buffer as PDF. Corresponding `rcd-paps-buffer-to-pdf-view`
command will open the PDF in PDF reader.

3. Command `M-x rcd-paps-region-to-pdf` is to generate PDF file for
selected region as PDF. Corresponding `rcd-paps-region-to-pdf-view`
command will open the PDF.

4. To setup header or footer, use `M-x rcd-paps-setup-header` or `M-x
rcd-paps-setup-footer` options. Use general
`rcd-paps-setup-header-and-footer` option.

5. There are many improvements to be done for this package, send your
suggestions to author.

== Source of rcd-paps.el

```lisp
 (file-to-string "/home/admin/Programming/emacs-lisp/rcd-paps.el") 
```



== Source of this page
[source,asciidoctor]
 ----
 (string-replace "\n----" "\n ----" (hyperscope-text hs::id)) 
 ----
````