AcroTeX.Net

The mkstmpdad Bundle

Using the mkstmp_pro Package to Create Custom Stamps

and

Using the aeb_dad Package to Create Drag and Drop Matching

D. P. Story

Prepared: November 12, 2012

www.acrotex.net
Version 1.0

Table of Contents

1. Introduction

This bundle consists of two related LaTeX packages:

mkstmp_pro is used to create stamps for display in Adobe Acrobat or Adobe Reader. Adobe Acrobat (and Adobe Distiller) are required to produce these stamps. The mkstmp_pro package requires aeb_pro, for it uses the `docassembly` environment to create the stamp file.

The contents of the stamp file appears in the Comment panel, under the Add Stamp tool. A stamp can be dragged and dropped into a PDF document and a comment can be attached to it.

aeb_dad is an application to the mkstmp_pro package; aeb_dad creates a "matching" game, in which the user drags a stamp and drops it into a target region. Underlying JavaScript then determines whether the user has dropped the stamp in the correct region or not. This is as close to a drag and drop feature that you can get using Adobe PDF.

The matching "game" created by the aeb_dad package works for users that have Adobe Acrobat, but the big news here is that it works also for users of Adobe Reader XI! In the version 11 release, Adobe Reader is now able to fill in forms and save and to provide access to all comment features without requiring Reader Extended PDF.[1]

For more information on stamps, I recommend the book *All About PDF Stamps, In Acrobat & Paperless Workflows*, by Thom Parker. See Thom's web site **WindJack Solutions** at windjack.com.

2. The mkstmp_pro Package

The mkstmp_pro enables the user of Acrobat Pro to *conveniently* create custom stamps.

2.1. Requirements and Installation

mkstmp_pro requires aeb_pro dated 2012/11/09 or later; in particular, there has been a change to the `aeb_pro.js` file, so this new file must be installed, following the steps described in the aeb_pro manual.

As for the installation of mkstmp_pro, if not automatically installed by a TeX system, just copy `mkstmp_pro.sty` into a folder named `aeb_dad`. If appropriate, refresh the filename database of your TeX system.

Examples and Documentation. Unzip `aeb_dad.zip` anywhere you want, outside the LaTeX search path. The ZIP file will create a folder named `aeb_dad`, containing program files, and folders `doc` and `examples`.

[1] This only applies to non-XFA documents. XFA still requires Reader Extensions to save filled in forms.

2.2. Testing the system

The mkstmp_pro package comes with an example file, uspres_stamps.tex, which is found in the aeb_dad > examples > mkstmp_pro folder. Accompanying this file is uspres.pdf, found in the images subfolder.

The verbatim listing of uspres_stamps.tex is given below.

```
1  \documentclass{article}
2  \usepackage[web=designi]{aeb_pro}
3  \usepackage{mkstmp_pro}
4  \pagestyle{empty}
5
6  \title{U. S. Presidents Stamps}
7  \author{D. P. Story}
8
9  \setStampPath{C:/Users/Public/Documents/%
10     My TeX Files/tex/latex/aeb/aebpro/mkstmpdad/%
11     examples/mkstmp_pro/images/uspres.pdf}
12 \makeStamps{%
13     {name=George Washington,page=0}
14     {name=John Adams,page=1}
15     {name=Thomas Jefferson,page=4}
16     {name=James Madison,page=3}
17     {name=John Quincy Adams,page=2}
18 }
19 \begin{docassembly}
20 \insertPreDocAssembly
21 \end{docassembly}
22
23 \begin{document}
24 \null\vfil
25 \begin{center}
26 \huge\sffamily\bfseries U. S. Presidents Stamps
27 \end{center}
28 \vfil
29 \end{document}
```

Before you can compile this file, you must edit lines (8)–(10) to match the path to the examples folder on your computer and save the changes. Follow the steps outlined in the next paragraph, in these general instructions, *my_stamps* refers to the demo stamp file uspres_stamps.

The following are the steps for creating a stamp file, *my_stamps*.

1. **Create the Stamp file.** Now, LaTeX *my_stamps*.tex, convert the DVI to PS using dvips, then convert to PDF using Adobe Distiller. If all works out, the file

my_stamps.pdf is produced. When the file first opens in Acrobat after distillation, there are document assembly methods that import the images of the presidents into the newly created file. Give it a second or two for the process to complete before saving *my_stamps*.pdf.

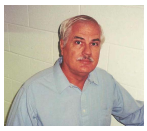
2. **Move the Stamp file.** After the stamp file is created, you need to move it to the stamp folder, where Acrobat expects it to be. To find this location, start Acrobat and open the JavaScript Debugger Window (Ctrl+J/Cmd+J), and type in the following line of code,

```
app.getPath("user","stamps");
```

   Place your mouse cursor on this line and press Ctrl+Enter/Cmd+Enter (or just use the Enter key on the keypad). Acrobat will execute this line and return the path to the user stamp folder. Navigate to this folder, and copy (or move) *my_stamps*.pdf to this folder.

3. Restart Acrobat.

4. On restart, the stamps your newly installed stamps should be visible: Open the Comment pane of Acrobat, and select the Add Stamp tool.



If you built and installed the uspres_stamp.pdf file correctly, you can use your stamps through the user interface of Acrobat, or reference them with the annot_pro package, as I did here (see the comments of George Washington in the margin).

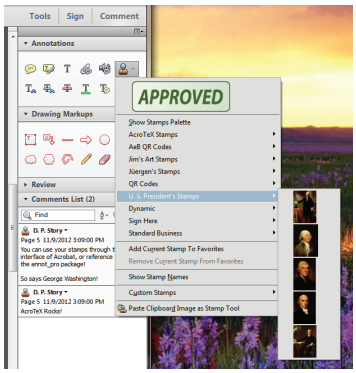


Figure 1: Stamp tool

➤ To use stamps through the user interface of Acrobat:

- For Acrobat 9 or earlier, select Comments > Show Comments & Markup. The Stamp tool can be seen on this toolbar. For Acrobat 10 and later, select the Comment panel on the right, again, the Stamp tool is revealed. Figure 1, page 5, show the stamp tool, with **U.S. Presidents Stamps** selected.

- Select a stamp, click on it, and bring your mouse back over the document. An image of the stamp should appear. Place it by left-clicking.
- Double click on the stamp to open the associated Pop-Up Note, used to associate a note with the stamp.

2.3. The details

A PDF file is used to create a stamp file. Each page that contains an image for a stamp must be a *page template* for that image to be recognized as a stamp. A page template has an associated *name* that is used as the *name* of the stamp, as can be seen through the user interface. The stamps that appear in a stamp file are listed under that *title* of the stamp file in the Stamp tool menu, listing all available stamps, see Figure 1, page 5.

In light of the above paragraph, describing a essentials of stamps, let's go through the listing of `uspres_stamps.tex`, which we'll reproduce in this page.

```
1  \documentclass{article}
2  \usepackage[web=designi]{aeb_pro}
3  \usepackage{mkstmp_pro}
4  \pagestyle{empty}
5
6  \title{U. S. Presidents Stamps}
7  \author{D. P. Story}
8
9  \setStampPath{C:/Users/Public/Documents/%
10  My TeX Files/tex/latex/aeb/aebpro/mkstmpdad/%
11  examples/mkstmp_pro/images/uspres.pdf}
12
13 \makeStamps{%
14   {name=George Washington,page=0}
15   {name=John Adams,page=1}
16   {name=Thomas Jefferson,page=4}
17   {name=James Madison,page=3}
18   {name=John Quincy Adams,page=2}
19 }
20
21 \begin{docassembly}
22 \insertPreDocAssembly
23 \end{docassembly}
24
25 \begin{document}
26 \null\vfil
27 \begin{center}\huge\sffamily\bfseries
28 U. S. Presidents Stamps
29 \end{center}
30 \vfil
31 \end{document}
```

Line (2) we use the aeb_pro package, which defines the `docassembly` environment, seen in lines (19)–(21). Line (3) inputs mkstmp_pro, this package.

On line (6) we title this document, the title appear in the Stamp menu, as seen in Figure 1 of page 5.

The `\setStampPath` command is defined in this package to point to the file containing the images to be made into stamps, call this the *images file*. It is an absolute path.

The `\makeStamps` command is the one that describes the images to be imported as stamps. Details are found below; in this example we specify a `name` for the stamp, and the `page` on which this stamp is to be found in the images file.

The `docassembly` environment, line (21) through line (23), encloses the command `\insertPreDocAssembly`. These two are defined in aeb_pro, but mkstmp_pro modifies `\insertPreDocAssembly` to import and name the images.

This content of the actual document is listed in lines (26)-(30). No DVI file is produced unless there is content. Here, we have a simple title page.

• Commands of mkstmp_pro

The mkstmp_pro only defines two commands, \setStampPath and \makeStamps.

- \setStampPath{absolute_path} defines the absolute path to the images file, the file that contains the images to be imported into the stamp file and used as stamps.
- \makeStamps takes a single argument that describe the images to be imported. The syntax is,

```
\makeStamps{%
    {name=name_1,page=page_1}
    {name=name_2,page=page_2}
    ...
    {name=name_n,page=page_n}
}
```

The images will appear in the stamp file in the same order they are listed. The value of the name key is the name to be associated with the stamp. The value of the page key is the page that this image is found on in the *image* file. (Notice that in the verbatim listing, the images are not imported in the same order they are listed in the images file.) There is another key, not shown above, called path. In theory, you can import an image in another file, different from the one declared by \setStampPath. It is perhaps best to have all images in a single file, however.

2.4. The mkstmp_pro workflow

The workflow comes in two steps, prepare an *images file*, create and install the *stamp file*.

• The images file

The mkstmp_pro package requires an *images file*, a file containing all the images to be made into stamps in the stamp file. For the uspres_stamps.tex file, the images file is uspres.pdf, as seen at the end the absolute path declared by \setStampPath.

Creating an images file is easy, given that you are using Acrobat. Combining files into a single PDF is accomplished by opening Acrobat and selecting Combine Files into PDF from the menu or toolbar. Now the good folks have moved things around over the years, so this features can appear in diverse location depending on the version you are using. Your Acrobat may have a Create button on the toolbar, as seen in Figure 2.[2]

The use of this feature is quite intuitive. Select the files you want to include in your images file. Combine and save them to the appropriate location. (Acrobat supports an enormous variety of file types, includes PDF, JPG, PNG, GIF, etc.)

[2] Version 9 of Acrobat has a Create button, but the feature here is now called Merge Files into a Single PDF. It is also found under File > Combine.

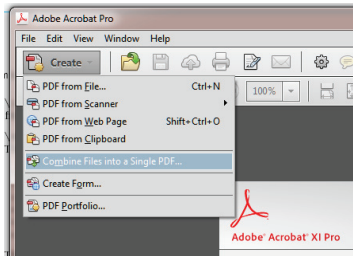


Figure 2: Combine Files into a Single PDF

• Create and install a stamp file

Create. To create a stamp file, take the sample file `upres_stamp.tex` and save it under a different name, say *`my_stamps`*`.tex`. Bring your newly created stamp file into your editor.

1. Set the `\title` and `\author` of *`my_stamps`* as desired. The `\title` will appear as a menu item of the Stamp tool.
2. Edit the argument of `\setStampPath` to point to the `image` file.
3. Edit the argument of `\makeStamps`. There should be one token for each if the stamp images you are importing. Follow the formatting of the sample file.
4. Compile *`my_stamps`*`.tex`, convert to PS (using dvips or dvipsone), and distill using Adobe Distiller. This creates your stamp file *`my_stamps`*`.pdf`.

Install. After the stamp file is created, move it to the stamp folder, where Acrobat expects it to be. To find this location, start Acrobat and open the JavaScript Debugger Window (`Ctrl+J`/`Cmd+J`), and type in the following line of code,

```
app.getPath("user","stamps");
```

Place your mouse cursor on this line and press `Ctrl+Enter`/`Cmd+Enter` (or just use the `Enter` key on the keypad). Acrobat will execute this line and return the path to the user stamp folder. Navigate to this folder, and copy/move *`my_stamps`*`.pdf` to this folder.

Restart Acrobat, if all went as planned, your new stamps should be listed in the menu listing of the Stamp tool. Verify this. If success, they are ready for use!

2.5. Using stamps with annot_pro

Earlier, on page 5, the George Washing stamp was used. The verbatim listing of that is given below.

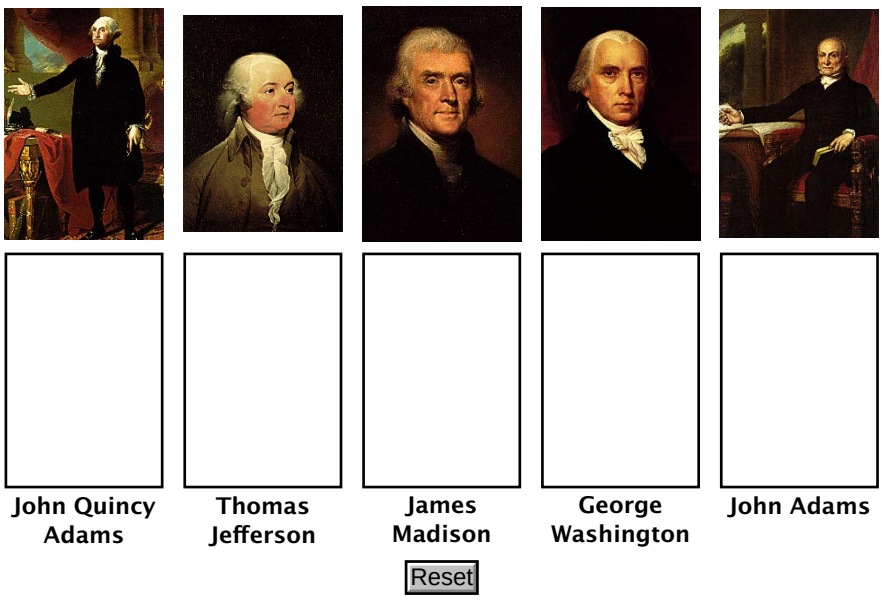
```
\annotpro[subject={AcroTeX makes stamps},title={D. P. Story},
type=stamp,name=\#George Washington]{You can use your
stamps through the user interface of Acrobat, or reference
them with the annot\_pro package!\n\n So says George
Washington!}
```

Note that the value of the `name` key is specified as `#George Washington`, not simply as `George Washington`. Custom stamp names require their name be prefixed with a `#`. The `mkstmp_pro` package automatically insert the required prefix, but not `annot_pro`.

See the documentation of `annot_pro` for more details on how to use the `\annotpro` command.

3. The aeb_dad Package

When Adobe Reader XI was released with its full support for comments (and saving form fields), I saw this opened up possibilities for using dragging and dropping stamps (DAD). Suppose you had a series of stamps, and a series of push buttons. Each button is associated with one of the stamps, but which one? The user, even using ARXI, drags and drops a stamp onto one of the push buttons. The underlying JavaScript determines whether it was the right choice, if now, the stamp is returned is starting position. Below is the example demonstrated in `dd_uspres.tex`, the demo file for this package.



Drag and drop the image of each President onto the corresponding rectangular region. Use your little gray cells, I am watching.

3.1. Requirements and Installation

The aeb_dad package requires annot_pro dated 2011/11/10 or later.

As for the installation of aeb_dad, if not automatically installed by a TEX system, just copy aeb_dad.sty into a folder named aeb_dad. If appropriate, refresh the filename database of your TEX system.

Examples and Documentation. Unzip aeb_dad.zip anywhere you want, outside the LATEX search path. The ZIP file will create a folder named aeb_dad, containing program files, and folders doc and examples.

3.2. Testing the system

The example file for this package is dd_upres.tex. Assuming you have installed and tested the Presidential stamps as per the instructions in the section titled 'Create and install a stamp file' on page 8, you are ready to compile the example file. After you compile, convert to PS, and distill using Adobe Distiller, you get the PDF file, dd_upres.pdf. *Don't forget to save the file*, this also saves the JavaScript that is imported into the document.

The behavior of the stamps and push buttons in the file dd_upres.pdf is the same as the Presidents drag and drop demonstration seen on the previous page.

3.3. The details

Let's look at the verbatim listing of dd_upres.tex.

```
\initDDGame{Presidents}
\ddDimens{%
    iconwidth=1in,iconheight=1.5in,
    targetwidth=1in,targetheight=1.5in}

\ddGameIcon{George Washington}\quad
...
\ddGameIcon{John Quincy Adams}

\ddTargetOfIcon{John Quincy Adams}
    {John Quincy Adams}\quad
...
\ddTargetOfIcon{George Washington}
    {George Washington}\quad
...
\ddReset
```

\initDDGame initializes the DAD game, each game must have a unique name.

\ddDimens sets up the dimensions of the stamps and the push buttons.

\ddGameIcon is a command that sets up the positioning of the stamps, the argument of this command is the *name* of the stamp to be used.

\ddTargetOfIcon is a comment that creates a push button that is a target of one of the stamps. The first argument is the *name* of the stamp, the second is the caption that appear beneath the push button.

\ddReset is a reset button. It puts the stamps back into their initial positions, and returns the push buttons to their original appearance.

The details follow in the order they appear in the document.

• Commands of aeb_dad

`\initDDGame{title}`

The `\initDDGame` command must appear on the same page on which the stamps and push buttons appear. The argument *title* is the title of the DAD game; typically, it should follow the same rules as a JavaScript variable, because if forms the base name of the push buttons and the reset button. `\initDDGame` defines a page option action that sorts through the stamps on the page for the ones that below to *title* and places them in an array.

`\ddDimens{key-value pairs}`

The key-values are,

- `iconwidth=`*width* is the width allotted to the icon (stamp). The default value is `\defaultStampWidth`, defined in annot_pro as 50bp.
- `iconheight=`*height* is the height allotted to the icon (stamp). The default value is `\defaultStampHeight`, defined in annot_pro as 50bp.
- `targetwidth=`*width* is the width of the target push button. The default value is 1.25in, as set by aeb_dad.
- `targetheight=`*height* is the height of the target push button. The default value is 1.25in, as set by aeb_dad.

`\ddGameIcon{icon_name}`

This is the command that places a stamp with name of *icon_name*. Use this command repeatedly with, of course, different names. Arrange them on the page as desired.

`\ddTargetOfIcon{icon_name}{caption}`

This command creates a target push button. There must be one `\ddTargetOfIcon` for each `\ddGameIcon`. The first argument, (icon_name) must match one and only one (icon_name) arugment of `\ddGameIcon`. (There must be a one-to-one correspondence between stamps and push buttons.) The second argument *caption* is a short caption that appear at the bottom of the button. This caption should be descriptive on the icon (stamp) that it matches with.

aeb_dad defines two other commands relatex to `\ddTargetOfIcon`.

```
\newcommand{\ddTargetCaption}[1]{\\[3pt]%
    \parbox[t]{\linewidth}{\centering\ddm@targetfmt#1}}
\newcommand{\ddTargetFmt}[1]{\def\ddm@targetfmt{#1}}
\ddTargetFmt{}
```

\ddTargetCaption sets the caption in a \parbox that is 3pt beneath the button. The command \ddm@targetfmt is a formatting command, it is defined by \ddTargetFmt. Use the command \ddTargetFmt to set the formatting (or styling) of the captions, it takes one argument which is passed to the tex stream prior to the caption. For example, for the President Stamp game appearing on page 5, we declared

```
\ddTargetFmt{\sffamily\small\bfseries}
```

The default is \ddTargetFmt{}, no styling.

\ddReset[*title*]

The \ddReset button restores the most recently defined DAD Matching Game to its original state. If, for some reason, the most recent one is not the one you want reset, use the optional argument *title* to pass the title of the game you want reset.

It is recommended that the reset button be always on the screen when the user is matching icons. The game executes a Field.setFocus() method to take the focus off of the stamps when they are dropped. The focus goes on the reset button. if the reset button is out of the user's viewing, AA or AR will scroll the page to place the reset button in the (middle of the) viewing area. This would not be a good experience for the user. For documents that are screen size, placement below the game is probably acceptable; for paper size pages, placement of the reset button between the row of stamps and the row of target buttons may work.

• Customizing the JavaScript

When a correct matching is made, the JavaScript function ddCorrectAction() is executed; and when an incorrect matching is made, ddWrongAction(). These two JavaScript functions simply place an alert in the screen with the message "Right!" or "Wrong", depending. These two messages, as well as a third one are defined by the following three commands.

```
\newcommand{\ddRightMsg}{"Right!"}
\newcommand{\ddWrongMsg}{"Wrong!"}
\newcommand{\ddDragOnlyOne}{"Drag one icon at a time"}
```

These may be redefined to other messages as desired.

The two functions ddCorrectAction() and ddWrongAction() first test for the presence of ddCustomCorrectAction() and ddCustomWrongAction(). If either one (or both) are defined, then it is these function that will be executed. The two JavaScript functions, then, are hooks into the system to modify how the game reacts to being right or wrong. See the second demo file dd_uspres_custom.tex for an example. The verbatim listing of the definitions of custom functions are given and discussed below. This example is just one of many possible definitions.

```
1  \begin{insDLJS}{dps}{Custom Notifications}
2  var aDADNames=new Array()
3  aDADNames["Presidents"] = "The Presidents of the United States";
4  function ddCustomCorrectAction(event,ddName) {
5      var page = event.target.page;
6      aDADCnt[ddName+page][0] += 1;
7      aDADCnt[ddName+page][1] += 1;
8      var nCorrect=aDADCnt[ddName+page][0];
9      var nTries = aDADCnt[ddName+page][1];
10     app.alert({cMsg: "That's right! Out of sight! ("
11         +nCorrect+" out of "+nTries+")",
12         nIcon: 3, cTitle: "AcroTeX Drag and Drop: "
13             +aDADNames[ddName]});
14 }
15 function ddCustomWrongAction(event,ddName) {
16     var page = event.target.page;
17     aDADCnt[ddName+page][1] += 1;
18     var nCorrect=aDADCnt[ddName+page][0];
19     var nTries = aDADCnt[ddName+page][1];
20     app.alert({cMsg: "Ding, Dong, that is Wrong! ("
21         +nCorrect+" out of "+nTries+")",
22         nIcon: 3, cTitle: "AcroTeX Drag and Drop: "
23             +aDADNames[ddName]});
24 }
```

- Lines (2) & (3) show how to get a custom title on the alert box, optional, just an idea.
- Lines (4)–(14) is the custom handler when the user has a correct matching.
  - There is a built in counter that tracks the number correct and the number of attempts. The counter array is shown in lines (6) & (7). The first one, line (6), keeps track of the number correct, and the other one, line (7), is the total attempts.

    The reset button `\ddReset` resets the counters to zero again.
  - Lines (10)–(13) is a custom alert box, that would say, for example, "That's Right! Out of sight! (2 out of 5)".
- Lines (15)–(24) shows the definition of `ddCustomWrongAction()`. It has a similar structure, but does not increment the number correct.

There are other ideas that come to mind. You can create a text field, and define your custom functions to write the score to it. Within the reset JavaScript function `resetDDM()`, a check is made for the presence of a function named `ddCustomResetAction()`; if it exists, it is executed as `ddCustomResetAction(page,ddName)`, where `page` is the page number on which the game resides, and `ddName` is the title of the game. You can use, for example, to then reset any text field you define to hold the score.

Now, I simply must get back to my retirement. ☺