AcroTeX.Net

The icon-appr Package

D. P. Story

Prepared: June 27, 2020

www.acrotex.net
Version 1.2.1, 2020/06/13

The icon-appr Package

Table of Contents

1. Introduction

In this package, we provide commands and methods for creating icon appearances for form field buttons, which includes push buttons, check box buttons, and radio buttons. Below are examples of the three types of buttons having icon appearances, rather than their customary appearances:

Push button

Check box

Radio buttons

Reset

For the check box and radio buttons, the Girl is 'off' and the Man is 'on'

The two sections that follow document the environment, commands, and methods for producing the above results. The above buttons are used in the demo files, these are found in the examples folder:

- `examples/icon-appr.exmpl.tex`
- `examples/pdfmark-drivers/icon-appr-pb.tex`
- `examples/pdfmark-drivers/icon-appr-eps.tex`
- `examples/pdfmark-drivers/icon-appr-eps-transp.tex`.

The first one listed above is for the pdflatex, lualatex, and xelatex drivers (applications), the latter three are designed for users of the dvips -> distiller workflow.

eforms package required

The eforms package, dated 2018/11/10 or later, is required to create buttons with icon appearances, this is because, as of this writing, the form fields produced by hyperref do not support the necessary markup to produce icon appearances.

1.1. What new: Version 1.2 (2020/06/05)

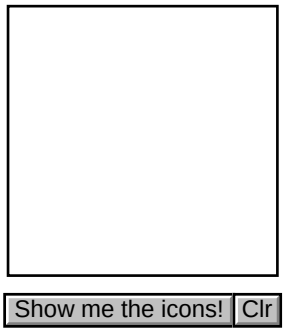
The basic functionality of this package is unchanged, as documented in subsequent sections. In this version, the **AP** entry is added to the **Names** dictionary of the PDF catalog. For this manual, the following code appears, new bits are highlighted in bold. The second line is the **Names** dictionary.

```
124 0 obj
<</AP 117 0 R/Dests 85 0 R/JavaScript 125 0 R>>
endobj
...
117 0 obj
<</Names[(girl)151 0 R(mani)137 0 R(scot)162 0 0 R]>>
endobj
```

The **AP** entry references the indirect object **Names** dictionary consisting of the icon names and their indirect references. The tricky part is that the names in this **Names**

array must be listed in alphabetical order. The datatool package is used for this purpose.[1]

This means that the names of the icons imported in the embedding environment are known to Acrobat/Adobe Reader. It also allows the icons to be manipulated using JavaScript methods; for example, use the button below to cycle through all icons in this document.



The underlying JavaScript of the push button uses the `this.getIcon(⟨icon-name⟩)` method. To use this method, the icons must be known, and now they are! All icons appearing in this document are EPS files, BION,[2] yet we can still manipulate their images using JavaScript. Adobe Reader supports `Doc.getIcon(⟨icon-name⟩)`. Open the JavaScript console, place your cursor on this.icons, and press Ctrl+Enter, Acrobat/Reader gives a readout of the icons known to this document. The above example is reproduced in `icon-appr-exmpl.tex` and `icon-appr-eps.tex`.

2. Methods for non-pdfmark drivers

pdflatex *lualatex* *xelatex*

The supported 'non-**pdfmark**' drivers are pdflatex, lualatex, and xelatex. To create icon appearances, embed the icon files with the `\embedIcon` command from within the `embedding` environment. This occurs in the preamble of the document.

```
\begin{embedding}
\embedIcon[⟨KV-pairs⟩]{⟨path⟩}
...
\end{embedding}
```

The name key

The `\embedIcon` embeds the icon file (⟨*path*⟩) in the document; it can then be referenced multiple times without significantly increasing the file size. The two relevant key-values (⟨*KV-pairs*⟩) are `name=⟨name⟩` and `hyopts={⟨various⟩}`. Internally, ⟨*name*⟩ is made into a control sequence (\⟨*name*⟩) which is used to reference the embedded icon file in the form field markup. Normally, ⟨*name*⟩ consists of letters, no active characters allowed; if ⟨*name*⟩ contains non-letters, its name may be referenced using the `\csOf` command (`\csOf{⟨name⟩}`). The other key-value pair is `hyopts={⟨various⟩}`,

The \csOf cmd

The hyopts key

[1] https://ctan.org/pkg/datatool

[2] Believe it or not

the value ⟨*various*⟩ are key-values of the `\includegraphics` command, which is used in the background. Passing any key-value through to `\includegraphics` may or may not have an effect. One useful key is the `page` key; when ⟨*path*⟩ leads to a multi-page PDF file, and `xelatex` is *not being used*, `page=`⟨*num*⟩ retrieves page ⟨*num*⟩ from the PDF document.

Example. We reproduce part of the file `icon-app-exmpl.tex`. First, in the preamble, embed all icon files to be used.

```
\begin{embedding}
\embedIcon[name=mani]{graphics/man1.pdf}
\embedIcon[name=girl]{graphics/girl.pdf}
\embedIcon[name=scot]{graphics/scot.pdf}
\end{embedding}
```

From these declarations, the commands `\mani`, `\girl`, and `\scot` are defined. Now in the body of the document, we create a push button:

```
\pushButton[%
  \TP{1}\BG{}\S{S}
  \I{\csOf{mani}}   % normal appearance, where we use \csOf to demonstrate its use
  \RI{\girl}        % rollover appearance, here, we reference the icon using \girl
  \IX{\scot}        % down appearance, we reference the icon using \scot
]{myButton}{50bp}{50bp}
```

The same techniques work for choice boxes and radio button fields. Refer to sample file `icon-appr-exmpl.tex` for a working example.

3. Methods for pdfmark drivers

dvips

For the **pdfmark** driver dvips, there are two techniques that have been developed. These techniques were developed because EPS files are the only graphics files dvips work with.

- **JavaScript approach:** Acrobat JavaScript has a method for embedding a number of graphics file formats as icons, which can then be used as icon appearance faces. This method requires the Acrobat application to open the newly created PDF file, after Distiller (or ps2pdf) has created the PDF file. Any supported graphics file format can be used. The method is explained in detail in Section 3.1.
- **Purely EPS approach:** We can use exclusively EPS files for icon appearances; in fact, the examples given in the **Introduction** section on page 3 were created by this method. Details are found in Section 3.2.

3.1. The JavaScript approach

aeb_pro package
Distiller or ps2pdf, and Acrobat

Requirements: The LaTeX package aeb_pro is required as it supplies the JavaScript code. Distiller or ps2pdf is used to create the PDF. Open the file in Acrobat where the JavaScript is executed to embed referenced files as icon objects and associate icon files with push button appearances.

push buttons only

This method only applies to push buttons, not to check box or radio button fields.

Again, the basic elements to use are the `embedding` environment in the preamble and the `\embedIcon` command.

```
\begin{embedding}
\embedIcon[⟨KV-pairs⟩]{⟨path⟩}
...
\end{embedding}
```

placement key
page key

The set of key-value pairs (⟨*KV-pairs*⟩) of `\embedIcon` is a little different than the ones listed in Section 2, these are (1) the `placement` key informs the underlying JavaScript where to place the icon file; (2) the `page` key can be used for multi-page PDF icons files to specify the number of the page to be used, this is a 0-based page number.

Syntax for the value of the `placement` key: The `placement` key "places" the image on the button faces of the field names listed (`myButton`). A push button has three appearance faces: normal appearance, rollover appearance, and down appearance. As a result of this, there is an optional argument that precedes the field name that determines the face of the button the icon is to appear on; the values are `[0]` (default, normal icon);[3] `[1]` (down icon); and `[2]` (rollover icon). The optional argument precedes the field name, and is shown in the example below. There must be no space between the optional argument and the field name; if you type '`[2]␣myButton`', for example, the field name is interpreted as '`␣myButton`', which is incorrect.

Example. This is a modified version of the example that appears in the sample file `icon-appr-pb.tex`. We begin by embedding the icon files in the document. The target field has name '`myButton`' and we place the images on it: `man1.pdf` is the normal appearance; `scot.gif` is the down appearance; and `girl.png` is the rollover appearance.

```
\begin{embedding}
\embedIcon[placement=myButton]{graphics/man1.pdf}
\embedIcon[placement={[1]myButton}]{graphics/scot.gif}
\embedIcon[placement={[2]myButton}]{graphics/girl.png}
\end{embedding}
```

Note the variety of icon file formats used.

In the body of the document, we create a push button. At the time the button is created, the icon files have not been imported or embedded, but we indicate that this button uses icon appearances by passing `\importIcons{y}` as an optional argument, *this is important.*

Important!

```
\pushButton[\BC{}\BG{}\S{S}\importIcons{y}
  \FB{true}\TP{1}]{myButton}{50bp}{50bp}
```

When the newly created PDF is first opened in Acrobat some JavaScript will execute and embed the icon files in the PDF, then populate the specified button faces with the specified icons.

By the way, a single `\embedIcon` command can provide multiple push buttons with its icon; the value of `placement` can be a comma-delimited list of field names (with optional argument preceding). For example,

[3] When no optional argument precedes the field name, it is understood to be the normal appearance.

```
\begin{embedding}
\embedIcon[placement={myButton,[1]myOtherButton}]{graphics/man1.pdf}
...
\end{embedding}
```

Refer to `icon-appr-pb.tex` for a working example.

3.2. The purely EPS approach

graphicxsp required

Requirements. The EPS method requires the use of the graphicxsp package, dated 2018/11/20 or later. Distiller or ps2pdf can be used to produce the PDF file; Acrobat is *not required* (unless another package requires it). It is strongly recommended that when creating check boxes or radio button fields that the newly created PDF be opened in Adobe Reader DC (or, optionally Acrobat itself) and *saved*. This will enable users of PDF-XChange Viewer/Editor to view these buttons correctly.

save the PDF!

The technique is similar to that of the non-**pdfmark** drivers. Again, we use the `embedding` environment in the preamble, but the `\embedEPS` command within the environment is used instead of `\embedIcon`.

```
\begin{embedding}
\embedEPS[⟨KV-pairs⟩]{⟨path⟩}
...
\end{embedding}
```

name key required

Where ⟨*path*⟩ points to an EPS file. The `name=`⟨*name*⟩ key-value is required in the optional argument ⟨*KV-pairs*⟩, other key-value pairs are passed to `\includegraphics`.

Internally, ⟨*name*⟩ is made into a control sequence (`\`⟨*name*⟩) which is used to reference the embedded icon file in the form field markup. Normally, ⟨*name*⟩ consists of letters, no active characters; if ⟨*name*⟩ contains non-letters, its name may be referenced using the `\csOf` command (`\csOf{`⟨*name*⟩`}`).

The \csOf cmd

Example. This is a modified version of the example that appears in the sample file `icon-appr-eps.tex`.

```
\begin{embedding}
\embedEPS[hiresbb,name=mani]{graphics/man1}
\embedEPS[hiresbb,name=girl]{graphics/girl}
\embedEPS[hiresbb,name=scot]{graphics/scot}
\end{embedding}
```

Here, we pass the `hiresbb` key to `\includegraphics`. From this declarations, the command `\mani`, `\girl`, and `\scot` are defined. Now in the body of the document, we create a push button:

```
\pushButton[%
  \TP{1}\BG{}\S{S}
  \I{\csOf{mani}}   % normal appearance, where we use \csOf to demonstrate its use
  \RI{\girl}        % rollover appearance, here, we reference the icon using \girl
  \IX{\scot}        % down appearance, we reference the icon using \scot
]{myButton}{50bp}{50bp}
```

Note that this is the same markup as was presented in Section 2. The same techniques work for choice boxes and radio button fields. Refer to sample file `icon-appr-eps.tex` for a working example.

When EPS methods are used, the Adobe transparency model can be used (Distiller required). See the sample file `icon-appr-eps-transp.tex`.

4. Parameters controlling the icon appearance for push buttons

The **MK** entry is used to provide an *appearance characteristic dictionary* containing additional information for constructing the annotation's appearance. The `eforms` package has key-value pairs that populates the **MK** dictionary; we describe the entries in the dictionary, these entries are entered through the optional argument of a `\pushButton` command. In the listing below, we give the key-value pairs, the first is the original key scheme, the second is the more user-friendly key. Additional details can be found in `eformman.pdf`, the documentation of the `eform` package.

I Indirect reference to the normal appearance of an icon. The keys used by eforms are `\I` and `normalappr`.

RI Indirect reference to the rollover appearance of an icon. The keys are `\RI` and `rollappr`.

IX Indirect reference to the down appearance of an icon. The keys are `\IX` and `downappr`.

Note. When importing icon appearances using JavaScript (Section 3.1), the above three keys are not used explicitly (JavaScript sets these key-value entries); use instead the key-value pair `\importIcons{y}`, as seen in the example given in Section 3.1.

IF The *icon fit dictionary*. The entries of the **IF** follow:

SW (name; optional) The circumstances under which the icon should be scaled inside the annot rectangle. The key is either `\SW` or `scalewhen`.

A always scale (the default value).
KVP: `\SW{A}` or `scalewhen=always`.

B Scale only when the icon is bigger than the annotation rectangle.
KVP: `\SW{B}` of `scalewhen=iconbig`.

S Scale only when the icon is smaller than the rectangle.
KVP: `\SW{S}` or `scalewhen=iconsmall`.

N Never Scale.
KVP: `\SW{N}` or `scalewhen=never`.

S (name; optional) The type of scaling to use the annot rectangle. The key to use is `\ST` or `scale`.

A *Anamorphic scaling:* Scale the icon to fill the annotation exactly, without regard to the original aspect ratio.
KVP: `\ST{A}` or `scale=nonproportional`.

P *Proportional scaling:* Scale the icon to fit the width or height of the rectangle while maintaining the icon's original aspect ratio (ratio width to height) (the default).
KVP: `\ST{P}` or `scale=proportional`.

A (array; optional) An array of two numbers between 0.0 and 1.0 indicating the fraction of the left over space to allocate at the left and bottom of the icon. A value of [0.0 0.0] positions the icon at the bottom-left corner; a value of [0.5 0.5] centers it within the rectangle. This entry is only used of the icon is scaled proportionally. The default is [0.5 0.5] the `annot` rectangle. The key is either `\PA` or `position`. The default is `\PA{.5 .5}` (no comma between numbers), in the user friendly style `position={.5 .5}` (no comma between numbers).

FB (Boolean; optional) If `true`, indicates that the button appearance should be scaled to fit fully within the bounds of the annotation without taking into consideration the line width of the border. The default is `false`. The key is `\FB` or `fitbounds`; the default is `\FB{false}` or `fitbounds=false`.

TP a code indicating position of text relative to icon. The key is either `\TP` or `layout`.

0 No icon; caption only. KVP: `\TP{0}` or `layout=labelonly`.

1 No caption; icon only. KVP: `\TP{1}` or `layout=icononly`.

2 Caption below icon. KVP: `\TP{2}` or `layout=icontop`.

3 Caption above icon. KVP: `\TP{3}` or `layout=iconbottom`.

4 Caption to the right of icon. KVP: `\TP{4}` or `layout=iconleft`.

5 Caption to the left of icon. KVP: `\TP{5}` or `layout=iconright`.

6 Caption overlaid on the icon. KVP: `\TP{6}` or `layout=labelover`.

That's it, now, back to my retirement!