qrcstamps: Create QR codes using stamps

D. P. Story
Email: dpstory@acrotex.net

processed June 2, 2018

Contents

```
⟨*package⟩
```

1 Description

This package attempts to create barcodes, at least the ones supported by Acrobat: PDF417, QR Code, Data Matrix. Acrobat has a barcode field that can be created in the standard way using eforms, but it does not work because the barcode renderer is never called to create the image for the barcode. Bummer.

The approach we take was suggested to me by Thom Parker, through personal communication, through his article on AcrobatUser.com and his book on PDF Stamps. The idea is to create *dynamic stamps* whose appearances are barcodes, such as a QR symbol.

In this version, we concentrate of the QR Code symbology. Whereas PDF417 and Data Matrix symbology may be created in the same way, there seems no need for either of these two. A QR Code symbology can be read by the scanners available on most mobile phones today. In that context, the symbology often contains an URL that can be scanned by a persons mobile, after which time, he (or she) may visit that web site without having to physically type in the URL address.

```
\RequirePackage{xkeyval}
```

2 Documentation and Code

`scandoc` The `scandoc` option calls some doc assembly JavaScript to scan the document for barcode stamps, this may not be necessary, depending on the version of Acrobat you have.

```
\DeclareOptionX{scandoc}{\let\grc@InputScanDoc\qr@scanDoc}
```

`!scandoc` During document development, you don't what to scan the newly created PDF after you build it to, perhaps, edit the text. In this case, use the `!scandoc` option. You want to scan the document when you build the document for the last time before publishing it. Be sure to save the document using the SaveAs menu item. This saves the stamp appearances (symbology in this case) as part of the document.

```
\DeclareOptionX{!scandoc}{\let\grc@InputScanDoc\relax}
```

The default is not to scan the document. Remember, the final document must be scanned; that is, it must be build with the `scandoc` option in effect.

```
\let\grc@InputScanDoc\relax
\def\qr@scanDoc{\InputIfFileExists{scandoc-grc.def}{\PackageInfo{qrcstamps}
   {Inputting the scandoc-grc.def file}}
   {\PackageWarning{qrcstamps}{Cannot find the file scandoc-grc.def}}}
```

`basename=`⟨*name*⟩ sets the base name of the barcode annotation. The form of the name is ⟨*basename*⟩_⟨*size*⟩. The default value of `basename` is `basename=AeBQRC`. This option need never be specified unless a package developer creates his own custom barcode fields and stamps. To reference these new stamps, the `basename` would have to be specified, unique that collection of stamps.

```
\DeclareOptionX{basename}{\def\QRBase{#1}}
\def\QRBase{AeBQRC}
```

```
\DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{insdljs}}
```

```
\ProcessOptionsX
```

The package used here is `annot_pro`, version dated 2017/06/06. I made a minor modification of `annot_pro` to accommodate `\qrcstamps`.

```
\RequirePackage{annot_pro}[2017/06/06]
\ifx\grc@InputScanDoc\relax\else\let\execjs=y\fi
```

2.1 The `\qrCode` command

We develop the stamp annotation for a QR barcode. The command `\qrCode`, defined below, uses `\annotpro` from `annot_pro`, and as such, its optional argument takes the same options as `\annotpro`. However, we add two new options, available only within the options list of `\qrCode`. modification of `annot_pro` to accommodate `\qrcstamps`.

`size` The `size` key takes one of three values `small`, `medium`, and `large`.

- `size=small` produces a stamp that holds at most 75 characters

- `size=medium` produces a stamp that holds at most 250 characters
- `size=small` produces a stamp that holds at most 500 characters

The choice of `size` should be the best fit for the data you provide `\qrCode`.

```
\define@choicekey+{annotprostampQR}{size}[\val\nr]{%
small,medium,large}[small]{\ifcase\nr\relax
    \def\apstamp@@size{Small}\def\qrc@def@W{1in}\or
    \def\apstamp@@size{Med}\def\qrc@def@W{1.5in}\or
    \def\apstamp@@size{Large}\def\qrc@def@W{2in}\else
    \def\apstamp@@size{Small}\def\qrc@def@W{1in}\fi}%
{\PackageWarning{qrcstamps}{Invalid choice of 'size=#1'\MessageBreak
    Permissible values are small, medium, large}}
\def\apstamp@@size{Small}
```

`allowresize=`⟨`true|false`⟩ The `allowresize` allows you to resize and move the stamp. By default, the stamp (qr code symbology) cannot be resized or moved.

```
\define@boolkey{annotprostampQR}{allowresize}[true]{}
```

`basename=`⟨*name*⟩ The `basename` key allows you to give a ⟨*name*⟩ to the stamp you want to use *locally*. It's unlikely you are using more than one barcode stamp collection, so this option is normally not used, but just in case, it is here. Specifying a value for the key `basename` overrides the choice for the option `basename`.

```
\define@key{annotprostampQR}{basename}[\QRBase]{\edef\QRBase{#1}}
```

`contents=`⟨*text*⟩ It's unlikely you'll want to include a text message as part of your stamp, but just in case, we previde the `contents` key.

```
\define@key{annotprostampQR}{contents}[]{\long\def\qrc@contents{#1}}
\let\qrc@contents\@empty
```

`\qrCode[`⟨*options*⟩`]{`⟨*content*⟩`}` The `\qrCode` command is the one that produces the QR Code stamp symbol, not as a barcode field, but as an annotation stamp. For printed material, the stamps are flattened and appear as you expect them. The ⟨*options*⟩ may be any option for a stamp annotation as produced by `\annotpro` plus the options `size` and `allowresize`. The required argument ⟨*content*⟩ is the text you want the qr barcode symbology to represent. Usually its an URL.

```
\newcommand\qrCode[2][]{\begingroup\def\n{\string\n}%
    \def\apstamp@@size{Small}\def\qrc@def@W{1in}%
    \setkeys*{annotprostampQR}{#1}%
    \annotpro*[widthTo=\qrc@def@W,#1,type=stamp,
    \ifKV@annotprostampQR@allowresize\else readonly\fi,
    title=QRC,subject={#2},%
    name=\#\QRBase_\apstamp@@size]{\qrc@contents}\endgroup}
```

```
⟨/package⟩
⟨*scandoc⟩
```

2.2 Document assembly

The document assembly code is executed when the `scandoc` option is used. It is executed then the document is first opened.

```
\begin{execJS}{scan4qrc}
var aBCStamps=new Array();
if (typeof scancomplete=="undefined") {
    var scancomplete=false,annots,isStamp,isHashtag;
    this.syncAnnotScan();
    for (var p=0; p<this.numPages; p++) {
         annots=this.getAnnots(p);
         if (annots!=null) {
            for (var i=0; i<annots.length; i++) {
```

At this time, we search only for annots of type `"Stamp"` and ones whose `AP` key has an entry beginning with '`#`'. Once found, we push it (the page number) onto the `aBCStamps` array.

```
                isStamp=(annots[i].type=="Stamp");
                isHashtag=(annots[i].AP.indexOf("\#")==0);
                if ( isStamp  &&  isHashtag ) {
                    aBCStamps.push(p);
                    break;
                }
            }
        }
    }
}
QRCscrollPage.index=0;
function QRCscrollPage() {
```

We go to the page where each stamp of the type we are interested in. Opening the page causes Acrobat to create an appearance for the stamp.

```
    if (QRCscrollPage.index<aBCStamps.length) {
         this.pageNum=aBCStamps[QRCscrollPage.index];
         QRCscrollPage.index+=1;
    } else {
        app.clearInterval(qrcTO);
        scancomplete=true;
        this.pageNum=0;
    }
}
if (aBCStamps.length>0)
    var qrcTO=app.setInterval("QRCscrollPage()", 5);
\end{execJS}
⟨/scandoc⟩
⟨*package⟩
```

Input the `scandoc-grc.def` code, if the option `scandoc` is in effect.

```
\grc@InputScanDoc
⟨/package⟩
```

3 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.